

Deterministic Edge Connectivity in Near-Linear Time*

Ken-ichi Kawarabayashi[†]

National Institute of Informatics, Tokyo, Japan
k_keniti@nii.ac.jp

Mikkel Thorup[‡]

University of Copenhagen
mikkel2thorup@gmail.com

December 4, 2015

Abstract

We present a deterministic algorithm that computes the edge-connectivity of a graph in near-linear time. This is for a simple undirected unweighted graph G with n vertices and m edges. This is the first $o(mn)$ time deterministic algorithm for the problem. Our algorithm is easily extended to find a concrete minimum edge-cut. In fact, we can construct the classic cactus representation of all minimum cuts in near-linear time.

The previous fastest deterministic algorithm by Gabow from STOC'91 took $\tilde{O}(m + \lambda^2 n)$, where λ is the edge connectivity, but λ can be as big as $n - 1$.

At STOC'96 Karger presented a randomized near linear time Monte Carlo algorithm for the minimum cut problem. As he points out, there is no better way of certifying the minimality of the returned cut than to use Gabow's slower deterministic algorithm and compare sizes.

Our main technical contribution is a near-linear time algorithm that contracts vertex sets of a simple input graph G with minimum degree δ , producing a multigraph \bar{G} with $\tilde{O}(m/\delta)$ edges which preserves all minimum cuts of G with at least two vertices on each side.

In our deterministic near-linear time algorithm, we will decompose the problem via low-conductance cuts found using PageRank a la Brin and Page (1998), as analyzed by Andersson, Chung, and Lang at FOCS'06. Normally such algorithms for low-conductance cuts are randomized Monte Carlo algorithms, because they rely on guessing a good start vertex. However, in our case, we have so much structure that no guessing is needed.

*Preliminary version [19] appears in the 47th ACM Symposium on Theory of Computing (STOC 2015).

[†]Ken-ichi Kawarabayashi's research is partly supported by JST ERATO Kawarabayashi Large Graph Project

[‡]Mikkel Thorup's research is partly supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career programme.

1 Introduction

In this paper we consider classic undirected graphs where the edges are a set of unordered pairs of vertices. We refer to them as a *simple graphs* to distinguish them from *multigraphs* (or pseudographs) allowing parallel edges. For both cases, the *edge-connectivity* is the smallest number of edges whose removal disconnects the graph. This is a classic global reliability measure for the connectivity of a graph. The set of edges removed are the *cut edges* of a (*global*) *minimum cut*, and the two components we get when removing them are the *sides of the cut*. We are here assuming that the graph is connected, which is trivially checked in linear time.

Our main result is a deterministic near-linear time algorithm to find the edge connectivity and a global minimum cut of a simple graph. It is based on a new understanding of the cuts in simple graphs that does not hold for multigraphs.

Previous work We will now discuss previous work on global min-cut algorithms. For the bounds we have n vertices, m edges, and (unknown) edge-connectivity λ . We consider both simple graphs and multigraphs. We will also consider *weighted graphs*, where edges have weights. Then edge-connectivity is no longer relevant, but the *size of a cut* is the total weight of the cut edges. For weighted graphs, parallel edges can be merged adding up the weights, so weighted graphs may be assumed simple.

In 1961, Gomory and Hu [10] showed that the global minimum cut problem can be solved computing $n - 1$ independent minimum s - t cuts, that is, cuts with s and t on different sides. They let s be an arbitrary vertex, and try with t being any of other vertices. The point is that to find a minimum cut, they just have to guess a vertex t on the side that s does not belongs to. The s - t cuts are understood via Menger's classic theorem [22]. We can thus use *any* s - t cut algorithm. On a multigraph, if we use the classic augmenting path algorithm of Ford and Fulkerson [6], we should work in parallel on all t , doing the same number of augmenting paths to each t . After λ augmentation rounds, for some t , we find an s - t cut of size λ which is also a global min-cut. The total time is $O(\lambda nm)$. We could also apply the $O(m^{3/2})$ time s - t min-cut algorithm of Even and Tarjan [5], and solve the global min-cut problem for multigraphs in $O(nm^{3/2})$ time. This improves Ford-Fulkerson when $\lambda = \omega(m^{1/2})$.

The first algorithm to compute a global minimum cut faster than n independent s - t cuts is the $O(\lambda n^2)$ time¹ algorithm of Podderyugin [25] for simple graphs from 1973. For many years, this algorithm did not receive attention until it was rediscovered by Karzanov and Timofeev [18] and by Matula [20], independently.

In the 1990s, the above bounds for simple graphs were generalized to multigraphs and weighted graphs. In 1990, Nagamochi and Ibaraki [23] gave an $O(m + \min\{\lambda n^2, pn + n^2 \log n\})$ time global min-cut algorithm for multigraphs where $p \leq m$ is the number of pairs of vertices between which the graphs has an edge. For weighted graphs, they got a general bound of $O(nm + n^2 \log n)$. Hao and Orlin [11] obtained an $O(nm \log(n^2/m))$ time algorithm for the directed weighted case. Stoer and Wagner [27] and Frank [7], independently, presented a very simple algorithm finding a global min-cut of an undirected weighted graph within the same $O(nm + n^2 \log n)$ time bound as in [23].

The current best deterministic algorithm for simple graphs is from 1991 due to Gabow [9] who gets down to $O(m + \lambda^2 n \log(n/\lambda))$ time. He also discuss multigraphs, with an implicit slightly worse bound of $O(m + \lambda^2 n \log n)$ Gabow [9, pp. 268-269]. A linear time $(2 + \varepsilon)$ -approximation of the edge-connectivity was presented by Matula [21].

¹We know $\lambda n = O(m)$, and this implies $\lambda n^2 = O(mn)$

All the above-mentioned algorithms have been deterministic. Randomized Monte Carlo (never sure about answer, see more details later) algorithms for the global minimum cut problem were initiated by Karger [14]. He first used a sampling technique to obtain an approximate global minimum cut in $O(m)$ time and an exact global minimum cut in $O(m\sqrt{\lambda})$ time. Karger and Stein [17] showed that random edge contraction works well for the global minimum cut problem, leading to an algorithm running in $O(n^2 \log^3 n)$ time. Finally, Karger [15] gave a randomized $O(m \log^3 n)$ time Monte Carlo algorithm for the problem.

For more detailed history for the global minimum cut problem, we refer the reader to the book by Schrijver [26]. We note that a deterministic near-linear time min-cut algorithm is known for planar graphs [3].

Main results As Karger [15] points out, there is no better way of certifying the minimality of the returned cut than to use Gabow’s slower deterministic algorithm [9]. Indeed, Karger’s algorithm is a *Monte Carlo* algorithm which gives the right answer with high probability but not with certainty. For many problems, we overcome this problem by either “certifying” the correctness of the output, or rerunning the algorithm, turning a *Monte Carlo* algorithm into a *Las Vegas* algorithm which guarantees that the output is correct, but takes long time with small probability. Unfortunately, we have no faster way of certifying a proposed minimum cut than computing one from scratch and comparing sizes.

In this paper, we present a deterministic near linear time algorithm for computing the edge connectivity and a global minimum cut for a simple graph. This is the first $o(mn)$ time deterministic algorithm for the problem. The previous best $\tilde{O}(m + \lambda^2 n)^2$ time bound of Gabow [9] is as good if λ is small, but we may have $\lambda = \Omega(n)$.

In near-linear time we can also compute the *cactus representation* of all global minimum cuts introduced in [4]. To do so we involve the previous fastest $\tilde{O}(\lambda m)$ time algorithm by Gabow [8] as a black-box. As for finding a simple minimum cut, we note here that Karger and Panigrahi [16] did give a near-linear time Monte Carlo algorithm for constructing the cactus data structure.

Technical Result Henceforth, we are only considering unweighted graphs. The min-cut algorithm we present is only for simple input graphs, but internally, it will also work with multigraphs.

By a *trivial cut*, we mean a cut where one side consists of a single vertex. Let δ be the minimum degree of a graph. Then δ is an upper bound on the edge-connectivity λ since it is the smallest size of a trivial cut. Finding δ is trivial, but we could have $\lambda < \delta$.

By Gabow’s result [9], we can find a global minimum cut in $O(\lambda m) = O(\delta m)$ time. Since we are aiming at $\tilde{O}(m)$ time, we may assume $\delta \geq \log^c n$ where c is an arbitrarily large constant. For our purposes, it will suffice with $c = 6$.

By *contracting a vertex set* $U \subseteq V$, we mean identifying the vertices in U while removing the edges between them. We may not check that U is connected, so this may not correspond to edge contractions. The identity of edges not removed are preserved. Our main technical contribution is to prove the following theorem:

Theorem 1 *Given a simple input graph G with n vertices, m edges, and minimum degree δ , in near-linear time, we can contract vertex sets producing a multigraph \overline{G} which has only $\overline{m} = \tilde{O}(m/\delta)$ edges and $\tilde{O}(n/\delta)$ vertices, yet which preserves all non-trivial min-cuts of G .*

From Theorem 1, we easily get our near-linear min-cut algorithm:

Corollary 2 *We can find a minimum cut of a simple graph G in near-linear time.*

² \tilde{O} hides polylog factor. Henceforth we refer $\tilde{O}(m)$ to near-linear.

Proof Let δ be the minimum degree of G . We apply the Theorem 1 to G producing the graph \overline{G} . We now run Gabow's min-cut algorithm [9] on \overline{G} , asking it to fail if the edge-connectivity is above δ . This takes $\tilde{O}(\delta \overline{m}) = \tilde{O}(m)$ time, and now we compare the output with the minimum degree δ . ■

Likewise, in near-linear time, we can obtain the cactus representation of all global minimum cuts from [4] by applying the cactus algorithm of Gabow [8] to \overline{G} . Having produced the cactus \overline{C} of \overline{G} , we just need to add min-degree vertices as extra needles so as to get the cactus of the input graph G . A description of this including the definition of the min-cut cactus is given in Section 6.

We note that Theorem 1 cannot hold if the input graph is a multigraph. To see this, consider a cycle of length $n \geq 4$, but where every edge is replaced by $k = (\log n)^{\omega(1)}$ parallel edges. Now every edge is involved in a non-trivial min-cut, and therefore no edges can be contracted. This shows that the contractions of Theorem 1 are very specific to simple graphs. Also, they can only preserve non-trivial min-cuts, for if we, for example, take a complete graph, then every edge is in a trivial min-cut.

While the reduction in Theorem 1 of the number of edges looks like a typical sparsification, it is not, for edges are contracted, not deleted, and the resulting \overline{G} will have much fewer vertices than G . In fact, we bootstrap Theorem 1 to prove a tighter theorem:

Theorem 3 *Given a simple input graph G with n vertices, m edges, and minimum degree δ , in near-linear time, we can contract vertex sets producing a multigraph \overline{G} which has only $\tilde{O}(n)$ edges and $\tilde{O}(n/\delta)$ vertices, yet which preserves all non-trivial min-cuts of G .*

Since the number of min-cuts in any multigraph is at most quadratic, we get

Corollary 4 *The number of non-trivial min-cuts in a simple graph with n vertices and minimum degree δ is at most $\tilde{O}((n/\delta)^2)$.*

We are not aware of anyone else that has observed that a large minimum degree in a simple graph implies few minimum cuts. It does appear that Corollary 4 could be derived from the cactus representation [4] that we also construct in near-linear time in this paper. However, with edge connectivity λ , it is straightforward (but messy and besides the main point) to generalize our algorithms to preserve all non-trivial cuts with less than $(2 - \Omega(1))\lambda$ edges, and there is no cactus representation for such approximately minimum cuts. This implies, for example, that our $\tilde{O}((n/\delta)^2)$ bound also holds for the number of non-trivial cuts with less than $3\lambda/2$ edges [12].

Minimum cuts and low conductance Our approach to finding a minimum cut involves cuts of low conductance, defined below. Generally we will define a cut by specifying one side $U \subset V$. Then the other side $T = V \setminus U$ is implicit. No side is allowed to be empty. Algorithmically, it will typically be the smaller side that we specify explicitly. The edges leaving U are the *cut edges*, and the set of cut edges is denoted $\partial U = \partial T$.

We are also interested in the number of edges with at least one endpoint in U called the *volume of U* defined as

$$\text{vol}(U) = \sum_{v \in U} d(v)$$

Now the *conductance of U* is defined by

$$\Phi(U) = \frac{|\partial U|}{\min\{\text{vol}(U), 2m - \text{vol}(U)\}} = \Phi(T).$$

Observation 5 *Let S be the smaller side of a min-cut of our simple graph G . Then either the cut is trivial with S consisting of a single vertex, or S has volume at least δ^2 and the conductance is $\Phi(S) \leq 1/\delta$.*

Proof The graph has minimum degree δ so the min-cut has at most δ edges. Since G is simple, a vertex $v \in S$ has at least $\delta + 1 - |S|$ edges leaving S . The total number of edges leaving S is thus at least $|S|(\delta + 1 - |S|)$, and for this to be at most δ , we need $|S| = 1$ or $|S| \geq \delta$. In the latter case, we have $\text{vol}(S) \geq \delta^2$, so $\Phi(S) \leq 1/\delta$. ■

Certify-or-cut In our algorithm, we are going to assume that the simple input graph G has minimum degree

$$\delta \geq (\lg n)^6.$$

By Observation 5, this means that any non-trivial min-cut has very low conductance. With this in mind, we are going to devise a near-linear time deterministic “certify-or-cut” algorithm that will either

1. Certify that there are no non-trivial min-cuts. In particular, this witnesses that any min-degree vertex forms the side of a global min-cut, or
2. Find a low-conductance cut.

We note that each of the above tasks alone is beyond our current understanding of deterministic algorithms. For the first certification task, recall the issue mentioned by Karger [15] that we have no efficient deterministic way of certifying that a proposed minimum cut is indeed minimum. Our task is no easier, for if it was, to certify that a cut of size $k \leq \delta$ is minimum, we could attach a complete graph on k vertices, where $k - 1$ of the vertices are new. Each new vertex defines a trivial cut of size $k - 1$, and the edge connectivity of the original graph is k if and only if there is no non-trivial minimum cut in the new graph.

For the second task, we want to find a low-conductance cut, e.g., using PageRank [2] as analyzed by Andersson, Chung, and Lang [1]. However, such algorithms for low-conductance cuts are randomized Monte Carlo algorithms, because they rely on guessing a good start vertex. For cut-or-witness, however, we only have to find a low conductance cut if we fail to witness the minimality of the trivial cuts, but then we will have so much structure that no guessing is needed.

Our certify-or-cut algorithm will illustrate some of the basic techniques presented in this paper, including a study of what happens in the endgame of PageRank when most mass has been distributed, yet some vertex is still left out.

The overall algorithm We will now sketch the basic ideas by using a more elaborate certify-or-cut algorithm for finding a minimum cut, and also point to the issues that arise.

Given a component C of subgraph H of G , suppose we can either

1. certify that C is a so-called “cluster” implying that no min-cut of G induces a non-trivial cut of C , or
2. find a cut of C of conductance $o(1/\log m)$.

Then, starting from $H = G$, we will recursively remove the low-conductance cuts, until we have a subgraph H of G where all the components are certified clusters. Inside these clusters we will identify a so-called “core” A with the property that no non-trivial min-cut of G makes any cut of A . Cores can therefore be contracted without affecting any non-trivial min-cut of G .

The important observation here is that when removing the low-conductance cuts, most edges survive in H . This is because we can amortize the edges removed over the edges incident to the smaller side where smaller is measured in terms of volume, that is, number of incident edges. Each edge incident to the smaller side pays $o(1/\log m)$ (because of the low-conductance cuts), and it can end on the smaller side at most $\lg m$ times, where $\lg = \log_2$. The total fraction of edges cut is thus $o(1)$, so most of edges remain when the cutting terminates.

We now point out the issues we have to address. The first issue is that as edges get removed, the degrees of the remaining vertices will decrease, and then the minimum degree could fall below $\lg n$, so we can no longer use Observation 5 to conclude that a non-trivial cut has conductance $o(1/\log m)$. Our fix to this issue will be to not only remove cut edges, but also “trim” the resulting components, removing all vertices that have lost $3/5$ of their original edges. As we shall see, this will only increase the number of edges removed by a factor 5, so most edges will still remain in the final clusters.

The second issue happens when we contract the cluster cores in a graph \overline{G} that preserves all the non-trivial min-cuts of G . This may introduce parallel edges, and hence Observation 5 fails completely, e.g., consider a path of length 4 where consecutive vertices are connected by δ parallel vertices. A non-trivial min-cut with two vertices on each side has conductance $1/2$. We will, however, argue that if a vertex is dominated by parallel edges, then it is somehow done and can be ignored.

Handling the above two complications will also force us to adopt a more complicated notion of a cluster, but our algorithm will still follow the basic pattern of the above sketch.

When done contracting cluster cores, \overline{G} will have only $\tilde{O}(m/\delta)$ edges, yet preserve all non-trivial min-cuts from G , as desired for Theorem 1. To find a minimum cut of \overline{G} , we finish by applying Gabow’s algorithm [9] as described in Corollary 2.

Contents. This paper is structured as follows. First we will show how to implement the certify-or-cut algorithm described above, since it introduces most of the interesting new ideas in a quite clean form. To do so, we will first describe our view of PageRank in Section 2, which includes a new theorem on the endgame. Next we describe the certify-or-cut algorithm in Section 3. After this, we will describe our new min-cut algorithm in Section 4, which involves many subtleties that will help us overcome the issues mentioned above. Finally, in Section 5 we prove the PageRank theorems claimed in Section 2. A cactus construction is given in Section 6.

2 Sparse cuts by PageRank

We are going to use the same PageRank algorithm as in [1]. We are operating with mass distributions $q \in \mathbb{R}_{\geq 0}^V$ assigning non-negative mass to the vertices. Given a subset U of the vertices, $q(U) = \sum_{v \in U} q(v)$ denotes the *total mass* on the subset. We refer to $q(U)/\text{vol}(U)$ as the *density* on U . For an individual vertex v , the density is $q(v)/d(v) = q(v)/\text{vol}(\{v\})$.

We start with some initial mass distribution $p^\circ \in \mathbb{R}^V$ on the vertices. Often we want the total mass to be 1, corresponding to a probability distribution.

The algorithm has a parameter α called the *teleportation* constant, and we assume $\alpha \leq 1/3$. The algorithm operates by moving mass between two mass distributions: a *residual mass* r which is initialized as the initial distribution p° , and a *settled mass* p which is initially zero on all vertices. Generally we say that the *density of mass* on a vertex is the mass divided by the degree

The algorithm works by *pushing* residual mass from vertices. To push the residual mass on u , we first settle a fraction α of the residual mass on u , and then we spread half the remaining residual mass evenly to

the neighbors of u . This is described in Algorithm 1. The overall algorithm is non-deterministic in that we

Algorithm 1: Push(α, u)

$p(u) \leftarrow p(u) + \alpha r(u);$
for $(u, v) \in E$ **do** $r(v) \leftarrow r(v) + (1 - \alpha)r(u)/(2d(u))$ $r(u) \leftarrow (1 - \alpha)r(u)/2.$

can apply pushes to the vertices in any order we want. To control the amount of work done, [1] introduces a parameter ε , and they only push from a vertex u if the *residual density* $r(u)/d(u)$ is at least ε . The resulting non-deterministic *approximate PageRank* algorithm³ is described in Algorithm 2. As noted in [1], the time

Algorithm 2: ApproximatePageRank($\alpha, \varepsilon, p^\circ$)

$r \leftarrow p^\circ; \quad p \leftarrow 0^V;$
while $\exists u : r(u)/d(u) \geq \varepsilon$ **do** Push(α, u)

to do a push at u is $d(u)$ and it settles $\alpha r(u) \geq \alpha d(u)\varepsilon$ of the residual mass. If we thus start with a total residual mass at most 1, the total amount of work is $O(1/(\alpha\varepsilon))$ [1, Theorem 1]. This does assume, however, that p° is presented in such a way that we have direct access to vertices density ε or more. In fact, we typically assume that the vertices are listed in order of non-increasing initial density. Then, for any ε , we find those with initial density above ε as a prefix of this list.

As ε approaches 0, the residual mass vanishes, and then, as proved in [1], the settled mass approaches a unique limit denoted $\text{PR}(\alpha, p^\circ)$ that we refer to as the *limit mass distribution*. The limit mass distribution will play an important role in our analysis, but algorithmically, we will only run the approximate PageRank from Algorithm 2 with $\varepsilon = 1/\log^{O(1)}$. In [1] they prove that

$$\text{PR}(\alpha, p^\circ) = p + \text{PR}(\alpha, r). \quad (1)$$

From [1, Proposition 2] we know that $\text{PR}(\alpha, \cdot)$ is a linear transformation $\mathbb{R}^n \rightarrow \mathbb{R}^n$ with no negative coefficients. For any $\sigma \in \mathbb{R}$, let $\bar{\sigma}$ be the distribution where all vertices have density σ . From [1, Proposition 1] we know that $\bar{\sigma}$ is a fix-point for $\text{PR}(\alpha, \cdot)$, that is, $\text{PR}(\alpha, \bar{\sigma}) = \bar{\sigma}$, and we call it a *stationary distribution*.

Mass can only be moved and settled via pushes. Consider an edge $(u, v) \in E$. Viewing it as directed from u to v , we get a positive flow when we push from u , pushing $(1 - \alpha)r(u)/(2d(u))$ mass over (u, v) to v while settling $\alpha r(u)$ mass at u . Likewise we get a negative flow over (u, v) when we push from v . Hence

Fact 6 *After any sequence of pushes for any $(u, v) \in E$, the total net flow of mass over (u, v) is $\frac{1-\alpha}{2\alpha} (p(u)/d(u) - p(v)/d(v))$.*

An important consequence is

Lemma 7 *If at some point all residual densities are bounded by σ , then from this point forward, the net flow over any edge is at most $\sigma/(2\alpha)$.*

Proof The point is that the residual distribution r is bounded by the stationary distribution $\bar{\sigma}$ with densities σ , so $\text{PR}(\alpha, r) \leq \text{PR}(\alpha, \bar{\sigma}) = \bar{\sigma}$. If q is a mass distribution settled from r , then $q \leq \text{PR}(\alpha, r) \leq \bar{\sigma}$, so $q(u)/d(u) - q(v)/d(v) \leq \sigma$ for every possible edge $(u, v) \in E$. By Fact 6, the net flow over (u, v) based

³If there is no confusion, we say “PageRank” for short.

on r is therefore at most $\sigma/2\alpha$. ■

We are going to find the side S of a low-conductance cut via a so-called “sweep” over the settled mass distributions p . To describe the sweep, as general notation, for any comparison operator $\circ \in \{=, <, >, \leq, \geq\}$ and $t \in \mathbb{R}$, define

$$V_{\circ t}^p = \{u \in V \mid p(u)/d(u) \circ t\},$$

e.g., $V_{\geq t}^p = \{u \in V \mid p(u)/d(u) \geq t\}$. Now let $\Phi(p)$ be the smallest conductance we can obtain by picking some threshold $\tau \in [0, 1]$, and considering the set of vertices with density at least τ , that is,

$$\Phi(p) = \min_{\tau \in [0, 1]} \Phi(V_{\geq \tau}^p).$$

To find $\Phi(p)$, we *sweep* over the vertices in order of non-increasing settled density. We only have to consider vertices with positive settled mass, including their incident edges, of which there are only $O(1/(\alpha\varepsilon))$ assuming that the total initial mass is 1. As described in [1], we can identify this cut in time $O((\log n)/(\alpha\varepsilon))$, and we shall further bring the time down to $O(1/(\alpha\varepsilon))$ in Section 5.1. The important question is: when does this sweep give us a cut of low conductance? We will answer this question in the next subsection.

2.1 Limit concentration and low conductance

We now state conditions under which approximate PageRank followed by a sweep yields a low conductance cut. The conditions are all based on the limit mass distribution p^* . We are interested in situations where the limit mass on set S deviates significantly from the average, as quantified by

$$\text{excess}(p^*, S) = p^*(S) - \text{vol}(S)/(2m).$$

The theorems below are proved using approximate PageRank and sweep. The first theorem is similar to results proved in [1].

Theorem 8 *Let be given an initial mass distribution p° of total mass 1, listing the mass for vertices in order of non-increasing density. Let $p^* = \text{PR}(\alpha, p^\circ)$. If there is a set S such that $\text{excess}(p^*, S) \geq \gamma$, then we can find a set T with $\text{vol}(T) \leq m$ and conductance*

$$\Phi(T) = O(\sqrt{(\alpha \log m)/\gamma}).$$

in time $O(\min\{m, \text{vol}(T)(\log m)\}/(\gamma\alpha))$. If no such a set S exists, we can report this in $O(m/(\gamma\alpha))$ time.

Given a bound $s \leq m\gamma/8$ on $\text{vol}(S)$, we find T in time $O(\min\{s, \text{vol}(T)(\log m)\}/(\gamma\alpha))$ with the additional guarantees that $\text{vol}(T) \leq 8s/\gamma$ and $\text{excess}(p^, T) \geq \gamma/(16 \lg(4s))$, or report in $O(s/(\gamma\alpha))$ time that there is no set S with $\text{vol}(S) \leq s$ and $\text{excess}(p^*, S) \geq \gamma$.*

The proof of Theorem 8 is deferred to Section 5. Without the running time, the first part follows directly from [1, Theorem 2], and indeed our Theorem 8 is the form they talk about informally in [1]. Unfortunately, when it comes to bound the running time, [1] only considers the start from a point distribution from a vertex that in a certain way is good in relation to a given cut. However, the running time in Theorem 8 is obtained using the same technique as in [1].

The endgame More interestingly, we study the endgame of the approximate PageRank algorithm, when after settling most of the mass, we discover that there is a vertex that even in the limit will not get enough density.

Theorem 9 *Let be given an initial mass distribution p° of total mass 1, listing the mass for vertices in order of non-increasing density. Let $p^* = \text{PR}(\alpha, p^\circ)$. If there is any vertex u with*

$$p^*(u)/d(u) \leq (1 - \gamma)/(2m),$$

then we can find a set T with $\text{vol}(T) \leq m$ and conductance

$$\Phi(T) = O(\sqrt{(\alpha \log m)/\gamma})$$

in time $O(m/(\gamma\alpha))$. In fact, we will obtain one of following cases:

- (i) $\text{excess}(p^*, T) \geq \gamma/(64 \lg(8m))$ and T is found in time $O(\min\{m, \text{vol}(T)(\log m)\}/(\gamma\alpha))$.
- (ii) T contains all small density vertices u with $p^*(u)/d(u) \leq (1 - \gamma)/(2m)$.
- (iii) A certification that there is no vertex u with $p^*(u)/d(u) \leq (1 - \gamma)/(2m)$.

We will not decide which case we get, but we will know which case we got.

The proof of Theorem 9 is deferred to Section 5. We note that if we just want a condition for finding a low-conductance cut, then Theorem 9 implies Theorem 8, for the overloaded set S in Theorem 8 implies that the average density outside S is $(1 - \Omega(\gamma))/(2m)$, and then Theorem 9 applies. We also note that Theorem 9 has a much stronger flavor than Theorem 8 in that Theorem 8 requires that the extra mass is a constant whereas Theorem 9 only requires that a single vertex is missing some mass. This asymmetry has to be there, for if we start with a point distribution s with mass 1 in a vertex u of minimal degree δ , then we always end up with $p(u) \geq \alpha$ corresponding to density $p(u)/d(u) \geq \alpha/\delta = \omega(1/m)$ if $\alpha = \omega(1/n)$, and yet there is no guarantees of a small conductance cut.

2.2 PageRank in our applications

In our applications, we are always going to use the same teleportation constant

$$\alpha_0 = 1/\lg^5 n.$$

Also, our initial distribution p° will almost always be obtained by distributing mass 1 evenly on some set X of vertices, that is, $p^\circ(v) = 1/|X|$ if $v \in X$; otherwise $p^\circ(v) = 0$ for $v \notin X$. We will simply say that we start *PageRank from* X , or from v if $X = \{v\}$. For simplicity, we will even let $\text{PR}(\alpha_0, X)$ denote the PageRank distribution $\text{PR}(\alpha_0, p^\circ)$, identifying in this case X with the distribution p° .

3 Certify-or-cut

In this section, using PageRank as described in Theorems 8 and 9, we will implement the “certify-or-cut” algorithm from the introduction, proving

Proposition 10 *Given a simple graph with minimum degree $\delta \geq \lg^6 n$, in near-linear time, we can either*

1. *certify that there are no non-trivial min-cuts, or*
2. *find a cut with conductance $o(1/\log m)$.*

Recall from the introduction that the point of the certify-or-cut is to illustrate our techniques in a simple form on a non-trivial problem. This is also why we will use the same parameters as in the rest of the paper even though the min-degree bound of Proposition 10 could easily be reduced. When we get to our real recursive min-cut algorithm, everything will become far more complicated.

3.1 Starting on the small side of a min-cut

Our first important observation is that if we start with a point mass on *any* vertex v on the small side S of a non-trivial min-cut, and the small side is not too large, e.g., $\text{vol}(S) \leq m/2$, then in the limit, we get a mass concentration on S so that Theorem 8 applies. This should be contrasted with the results from [1] which say that if S is a side of a low conductance cut, then a large fraction of the vertices can be used as starting points leading to mass concentration. In [1] they have to guess such a good starting vertex, resulting in a randomized algorithm. However, in our min-cut case, any vertex in S will do, which is why we have a chance of a deterministic algorithm.

Note that since S is a min-cut, v can have at most half its edges leaving S , for otherwise $S \setminus \{v\}$ would have a smaller cut around it. The result therefore follows from the following more general lemma.

Lemma 11 *Given a size bound s and a vertex $v \in V$, in $\tilde{O}(s)$ time, we can find a cut with conductance $o(1/\log m)$ if there is a min-cut side S with $\text{vol}(S) \leq s$ that contains v and a fraction $\varepsilon = s/(2m) + \Omega(1)$ of its neighbors. The condition is satisfied for every vertex $v \in S$ if $1 < \text{vol}(S) \leq s \leq m/2$.*

Proof We start PageRank with a point mass of 1 on v . Then we repeatedly push mass from v until its residual mass $r(v)$ is at most $1/\delta$. The mass from v will be spread evenly to its neighbors, so at the end, we have more than ε mass staying in S . Moreover, the residual mass on any vertex is now bounded by $1/\delta$. Next we apply the following lemma with $\mu = 1/\delta$:

Lemma 12 *If at some point the residual mass on every vertex is bounded by μ , then from this point forward, at most $\mu/(2\alpha_0)$ mass can move across any specific min-cut.*

Proof Since the minimum degree is δ , the maximal residual density is bounded by μ/δ . By Lemma 7, from this point forward, the net flow over any edge is at most $\mu/(2\alpha_0\delta)$. A min-cut has at most δ edges, so the net flow across any min-cut is therefore at most $\mu/(2\alpha_0)$. ■

After pushing the residual mass from its starting point v , by Lemma 12, the mass leaving S is at most $1/(2\alpha_0\delta) = o(1)$ since $\alpha_0 = 1/(\lg n)^5$ while $\delta \geq (\lg n)^6$. Thus, in the limit, the mass staying in S is $\varepsilon - o(1) = s/(2m) + \Omega(1)$, so $\text{excess}(p^*, S) = \gamma = \Theta(1)$. By Theorem 8, we now get a set T with

$$\Phi(T) = O(\sqrt{\alpha_0 \log m}) = o(1/\log m).$$

in time $O(s/\alpha_0) = \tilde{O}(s)$. This time bound is immediate from Theorem 8 with a bound $s \leq m\gamma/8$, but otherwise $s > m\gamma/8 = \Omega(m)$, and then the general time bound is $O(m/(\gamma\alpha_0)) = \tilde{O}(s)$, as desired.

Since every vertex v has at least half its neighbors on its side S of a non-trivial min-cut, the conditions of the lemma are satisfied if $1 < \text{vol}(S) \leq m/2$. ■

3.2 Balanced min-cut

We now consider the situation where both sides of some specific min-cut have volume at least $m/2$. We claim that there are less than 16 vertices that we can start from and we do not find a low-conductance cut. There are at most 2δ end-points of the cut edges, so there are less than 16 vertices incident to more than $\delta/8$ cut edges. These are the only bad vertices. Any other vertex v has at least a fraction $\varepsilon = 7/8$ of its neighbors on its side S of the min-cut. Moreover $\text{vol}(S) \leq s = 3m/2$, so $\varepsilon = s/(2m) + 1/8$. Thus, if we apply Lemma 11 to a vertex v , in $\tilde{O}(s) = \tilde{O}(m)$ time, we either find a cut of conductance $o(1/\log m)$, or conclude that v is bad. We run from 16 vertices. If they are all bad, we conclude that there is no min-cut where both sides have volume at least $m/2$. All this takes near-linear time, so to finish the proof of Proposition 10, it suffices to look for non-trivial min-cuts where the small side S has $\text{vol}(S) \leq m/2$.

3.3 Finding any min-cut using the endgame

We will now assume that we have a bound $s \leq m/2$ on volume of the small side of any min-cut. If there is a min-cut where one side has volume between $s/2$ and s , then we will find a sparse cut. We are only interested in non-trivial min-cuts. By Observation 5, the smaller side has volume at least δ^2 , so we will consider $s = m/2^i$ for $i = 1, \dots, \lceil \lg(m/\delta^2) \rceil$. For a given s , consider a min-cut $(S, V \setminus S)$ where $s/2 \leq \text{vol}(S) \leq s$. We will either find a low-conductance cut, or falsify the existence of $(S, V \setminus S)$.

We pick an arbitrary set U of $4m/(\alpha_0 s) = \tilde{O}(m/s)$ vertices. For each $v \in U$, we apply Lemma 11, either finding a desired low-conductance cut, or determining that v is not in S . The check from v takes $\tilde{O}(s)$ time, so checking all $v \in U$ takes $\tilde{O}(m)$ time. We now know that U is contained in the big side $V \setminus S$ of our min-cut.

Next we create a starting distribution, spreading mass 1 evenly on the vertices in U , thus giving each of them with an initial mass of $\mu = \alpha_0 s/(4m)$. None of this mass is in S , and by Lemma 12, the total mass that can move into S is at most $\mu/(2\alpha_0) = s/(8m)$, bounding the limit mass on S . Since $\text{vol}(S) \geq s/2$, the average limit density on S is thus at most $1/(4m)$. It follows that some vertex in S has limit density at most $1/(4m)$. This is the endgame considered in Theorem 9. In $\tilde{O}(m/\alpha_0)$ time, it finds a cut with conductance $O(\sqrt{\alpha_0 \log m}) = o(1/\log m)$. Otherwise we conclude that S did not exist.

For each of the logarithmic number of values of s , we thus spend near-linear time, so our total time bound is near-linear. If no low-conductance cuts are found, we conclude that there is no non-trivial min-cuts. This completes the proof of Proposition 10.

4 The min-cut algorithm

The reader may at this point want to review the sketch of our deterministic min-cut algorithm from the end of the introduction. The pseudo-code for the real algorithm is found in Algorithm 3. The different elements of the algorithm will be explained below. The basic idea is to construct a multigraph \overline{G} from G by contracting vertex sets while preserving all non-trivial min-cuts of G . The edge connectivity of G is at most δ , so if the edge connectivity of \overline{G} becomes bigger than δ , then there cannot be any non-trivial min-cuts in G , and then we can contract \overline{G} to a single vertex.

We note that if there are more than δ parallel edges between vertices u and v , then we can trivially contract $\{u, v\}$. There are therefore never more than δ parallel edges between two vertices in \overline{G} .

When a vertex set is contracted to a single vertex, we call it a *super vertex* while the original vertices from G are called *regular vertices*. If we just say a vertex it can be of either kind. The degrees of the regular

vertices (including multiple edges we created during our algorithm) does not decrease, so they will always have degrees at least δ .

Algorithm 3: Min-cut(G). Here G is a simple graph with m edges and minimum degree δ

```

if  $\delta \leq \lg^5 m$  then
  | find min-cut in  $G$  using Gabow's algorithm [9].
 $\overline{G} \leftarrow G$ ; //  $\overline{G}$  preserves non-trivial min-cuts
repeat
  |  $H \leftarrow \overline{G}$ ;
  | Remove passive super vertices from  $H$  and trim  $H$ ; // Section 4.1 and 4.3
  | while some component  $C$  of  $H$  is not known to be a cluster do
  |   | Find cut  $(A, B)$  of  $C$  with conductance  $\leq \Phi_0 = 1/(20 \lg m)$ ; // Sections 4.5-4.11
  |   | Remove cut edges from  $H$  and trim  $H$ 
  | Take each cluster component of  $H$  and contract its core to a super vertex in  $\overline{G}$ ; // Section 4.2
  | // Contracts 1/2 the edges in  $\overline{G}$  by Lemma 18 in Section 4.4
until  $\geq 1/20$  of edges in  $\overline{G}$  incident to passive super vertices;
//  $\tilde{O}(m/\delta)$  edges left in  $\overline{G}$  by Lemma 16 in Section 4.3
Find a min-cut in  $\overline{G}$  using Gabow's algorithm [9];

```

4.1 Clusters

Our min-cut algorithm is centered around finding clusters in \overline{G} defined below.

First, a set $C \subseteq V$ of vertices is called *trimmed* if for each $v \in C$, at least $2/5$ of the edges from v in \overline{G} stay in C . The set C is called a *cluster* if it is trimmed and for every cut of size at most δ in \overline{G} , one side contains at most two regular vertices and no super vertices from C .

Note that if a trimmed vertex set C only consists of regular vertices, then any one of them has at least $2\delta/5$ neighbors in C , so C has at least this many vertices. Thus, if C is a cluster, it is always clear which *side of a min-cut C belongs*; namely the side with the super vertices if any; otherwise the side with almost all the regular vertices.

The condition of having all but at most two regular vertices from C on the same side of any min-cut may seem a bit ad-hoc, but we have the following lemma stating how more than two makes a big difference.

Lemma 13 *Consider a trimmed vertex set C and a cut (T, U) of \overline{G} of size at most δ . If $T \cap C$ has no super vertices and at least 3 regular vertices, then $T \cap C$ has at least $\delta/3$ regular vertices.*

Proof The proof is very similar to that of Observation 5. Consider $T \cap C$ which has no super vertices. Since C is trimmed, the internal degree of regular vertices in C is at least $2\delta/5$, so the number of edges crossing from $T \cap C$ to $U \cap C$ is at least $|C \cap T|(2\delta/5 + 1 - |C \cap T|)$, but we have at most δ cut edges, so we conclude that $|C \cap T| \leq 2$ or $|C \cap T| \geq 2\delta/5 - 1 > \delta/3$. ■

4.2 Contracting the cores

The goal of our algorithm will be to find a family \mathcal{C} of non-overlapping clusters such that the number of edges not internal to clusters is $\overline{m} = \tilde{O}(m/\delta)$. Identifying a core of each cluster, defined below, we will

produce a graph \overline{G} with $O(\overline{m})$ edges, yet preserve all non-trivial cuts of size at most δ . We can then apply Gabow's algorithm [9], and find a minimum cut in $\tilde{O}(\overline{m}\delta) = \tilde{O}(m)$ time.

Note that because the clusters in \mathcal{C} are required to be non-overlapping, identifying a subset of vertices in one cluster will not stop any other cluster from being a cluster.

Consider a cluster C . We say a vertex $v \in C$ is *loose* if it is regular and at least $d(v)/2 - 1$ of its edges leave C (note that, by definition, at least $2/5$ of the edges from v stays in C). Let A be the set of vertices in C that are not loose. If more than $1/4$ of the edges incident to C are internal to A then we define A to be the *core* of C ; otherwise the core of C is empty (and contracting an empty core has no effect).

Lemma 14 *If a non-trivial min-cut of G has survived in \overline{G} , then it will also survive when we contract the core of any cluster in \overline{G} .*

Proof First we note that if a non-trivial min-cut of G survives in \overline{G} , then it must also be a min-cut (T, U) of \overline{G} . It was a min-cut of G , so it has $\lambda \leq \delta$ cut edges. Also, because it was a non-trivial cut in G with at least two vertices on each side, we must have at least two regular vertices or one super vertex both in T and in U .

We now consider a cluster C in \overline{G} with a non-empty core. Since (T, U) has at most δ cut edges, by the definition of a cluster, one side, say T , has at most two regular vertices and no super vertices from C . We will argue that these vertices in $C \cap T$ must be loose, hence that the vertices identified by the contraction of the core are all in U , for then this contraction preserves (T, U) .

Let v be one of the vertices from $C \cap T$, and assume for a contradiction that v is not loose. We will prove that we get a smaller cut by moving v to U , contradicting that (T, U) was a minimum cut. Since v is regular and both sides have at least one super vertex or two regular vertices, v is not the only vertex in T . Therefore we still have a cut after moving v to U .

Moving v only affects the cutting of edges incident to v . When v is in T , we cut all edges from v to C , except possibly one to another regular vertex in $C \cap T$. Since v is not loose, it has more than $d(v)/2 + 1$ edges from v into C , so with v in T , we cut more than $d(v)/2$ edges incident to v . Moving v to U , we stop cutting these edges, so we cut less than $d(v)/2$ edges incident to v , contradicting that (T, U) was a min-cut. ■

Lemma 15 *If a cluster C has k edges leaving it, then there are less than $3k$ edges incident to C that are not internal to the core. In particular, if the core is empty, we have $\text{vol}(C) < 3k$.*

Proof First we remove all loose vertices getting down to a vertex set A . We claim that at most $(2 + o(1))k$ of the edges incident to C are not internal to A .

Let ℓ be the number of edges leaving C from loose vertices. Then we have $k - \ell$ edges leaving C from vertices in A . Other edges incident to C but not internal to A are all incident to loose vertices.

Consider any loose vertex v in C . It has at least $d(v)/2 - 1$ incident edges leaving C and at most $d(v)/2 + 1$ edges staying in C . Loose vertices are regular, so $d(v) \geq \delta = \omega(1)$. It follows that the total number of edges incident to loose vertices is at most $(2 + o(1))\ell$. Therefore, the total number of edges not internal to A is at most $(2 + o(1))\ell + (k - \ell) \leq (2 + o(1))k$. This proves the lemma unless the core becomes empty.

The core becomes empty if and only if at most $1/4$ of the edges incident to C are internal to A , but this implies that the number of edges internal to A is at most $1/3$ of the number of edges not internal to A . Thus, if A is not the core, there are at most $(2 + o(1))k/3$ edges internal to A , and then we have at most

$(2\frac{2}{3} + o(1))k < 3k$ edges incident to C . ■

4.3 Active and passive super vertices

We say that a super vertex is *active* if it has at least

$$\delta^* = (\lg n)\delta/\alpha_0$$

incident edges; otherwise we call it *passive*.

The point in the high degree of an active super vertex is that no more than a fraction $\alpha_0/(\lg n)$ of the edges can go to a single neighbor. This will help us spread mass on one side of a min-cut in a way similar to what was described in Section 3.1. The point in the low degrees of passive super vertices is the following good bound on the total number of incident edges:

Lemma 16 *The total number of edges leaving passive super vertices is $\tilde{O}(m/\delta)$.*

Proof Consider the first cluster C with a non-empty core A that get contracted into a super vertex v^* . By *first* we mean that A itself does not have super vertices. Since A is non-empty, only loose vertices from C are not in A , and loose vertices are regular, so all vertices in C are regular. But C is also trimmed, so any vertex $v \in C$, has at least $2/5$ of its incident edges staying in C , and they all go to distinct neighbors since C has no super vertices. Thus $|C| \geq 2\delta/5$, and hence we have at least $2\delta^2/5$ edge end-points in C , corresponding to at least $\delta^2/5$ distinct edges. By definition of a non-empty core, this implies that A has at least $\delta^2/20$ internal edges that all get contracted into v^* . Now v^* may later be contracted with other vertices, but this can only increase the number of edges contracted in v^* . When v^* is passive, only δ^* edges leave v^* , which is at most a fraction $\delta^*/(\delta^2/20) = 20(\lg n)/(\alpha_0\delta) = \tilde{O}(1/\delta)$ compared to those contracted in v^* , and this holds for every passive super vertex. ■

Our algorithm will terminate successfully if the total number of edges in \overline{G} is less than 20 times the number of edges incident to passive super vertices, for then, by Lemma 16, we have only $\tilde{O}(m/\delta)$ edges in \overline{G} , and then, as described in Section 4.2, we can find a min-cut of G in near-linear time.

4.4 Cut, trim, shave, and scrap

Our algorithm generally works by alternation between cutting edges of a subgraph H of \overline{G} and trimming the resulting components of H as described below. We start with $H = \overline{G}$, and the process does not change \overline{G} . By *cutting* we refer to two cases. One is where we cut out a passive super vertex, removing its incident edges. The other is where we remove the edges of a low-conductance cut. By *trimming* we mean removing any vertex v from H that has lost more than $3/5$ of the edges it has in \overline{G} . When removing v , we also remove all its incident edges from H . When no more trimming is possible, each remaining vertex in H satisfies $d_H(v) \geq 2d_{\overline{G}}(v)/5$ which means that the components are trimmed as defined in Section 4.1.

The process will terminate when we somehow know that all remaining components in H are clusters in \overline{G} . Then we *shave* off the loose regular vertices v that have lost $d(v)/2 - 1$ of their incident edges. Let A be what is left of C . If less than $1/4$ of the edges incident to C are internal to A , we *scrap* A so that nothing remains from C . Otherwise A is a core that we contract it in \overline{G} . We note that while trimming and shaving are very similar, it is only trimming that can be done recursively. If the shaving was done recursively, we could easily end up losing all the edges in the graph.

We want to bound the number of edges cut, trimmed, and scrapped from H , for these are the edges that remain in \overline{G} when the cores are contracted.

Lemma 17 *If the total number of edges cut is c , then the total number of edges lost due to trimming, shaving, and scrapping is at most $4c$.*

Proof The proof is by amortization. The “lost degree” of a vertex $v \in H$ is the number of incident edges in \overline{G} that are not in H . We are interested in the total lost degree over all vertices in H , and it starts at 0 when $H = \overline{G}$. When we cut an edge, the total lost degree increases by 2. When we trim a vertex v , its lost degree was at least $3d_{\overline{G}}(v)/5$. Since v is removed, its lost degree is saved. On the other hand, we take out its at most $2d_{\overline{G}}(v)/5$ remaining incident edges, each increasing the lost degree of its end-point by 1. All together, by trimming v we reduce the total lost degree by at least $d_{\overline{G}}(v)/5$. Thus the total number of edges trimmed is at most twice the decrease in the total lost degree. The total increase by cutting is $2c$ and if the total lost degree is d after all the trimming is done, then the total number of trimmed edges is at most $2(2c - d)$.

It remains is to shave each cluster C down to the core, which is scrapped if too little, in which case the final core is empty. By Lemma 15, if C has k edges leaving, then at most $3k$ edges from C will end up removed because they are not internal to the final core. However, the k edges leaving C were already taken out, so we take out at most $2k$ additional edges. Thus, with a total of d edges leaving clusters after cutting and trimming, the last part takes out at most $2d$ edges. All in all, the trimming, shaving, and removal of undersized cores, takes out at most $2(2c - d) + 2d \leq 4c$ edges. ■

As mentioned above, we start the round with $H = \overline{G}$. As described at the end of Section 4.3, we are done if more than a fraction $1/20$ of the edges are incident to passive super vertices. Otherwise, we cut all edges incident to super vertices, and trim the resulting components.

Next we are repeatedly going to cut and trim using cuts of components of H of conductance at most

$$\Phi_0 = 1/(20 \lg m).$$

This is what we henceforth regard as a “low-conductance” cut. Later sections will prove that low-conductance cuts can be found efficiently if a component is not a cluster.

We claim that the total number of edges cut this way is at most a fraction $1/20$ of the edges in \overline{G} . The point is that the number of edges cut is a fraction $1/(20 \lg m)$ of the volume of the small side, and the same vertex can end on the smaller side only $\lg m$ times. Here size is measured by volume, that is, number of incident edges.

Including the at most $1/20$ of the edges of \overline{G} incident to passive vertices, we thus cut at most a fraction $1/10$ of the edges in \overline{G} . Hence, by Lemma 17, in total, we lose at most $1/2$ of the edges in \overline{G} . Summing up,

Lemma 18 *Cutting edges around passive vertices and edges of low-conductance cuts, trimming, shaving, and scrapping, leaves at least half the edges of \overline{G} in the resulting cluster cores of H .*

4.5 Cutting into clusters

We will now, at a high level, describe the process that repeatedly takes a component C of H , cut the edges of a low-conductance cut and trim the sides, stopping only when all remaining components are clusters. We start with a graph H with no passive super vertices, and the total time will be near-linear in the number of edges.

We need a measure for how close components are at being clusters. We generally say that a component C of H is s -strong if every cut (T, U) of \overline{G} with at most δ cut edges has $\min\{\text{vol}_C(T \cap C), \text{vol}_C(U \cap C)\} < s$. Note that C must always be $m(C)$ -strong. A very important part of this definition is that it is inherited by subgraphs, that is, if A is a subgraph of C and C is s -strong, then so is A . Being s -strong is thus preserved as we cut and trim. Let

$$s_0 = 64\delta/\alpha_0$$

Our goal will be to partition H into s_0 -strong trimmed components, for they are then all clusters:

Lemma 19 *If a trimmed component C of H is s_0 -strong, then C is a cluster.*

Proof If C is not a cluster, then there is a cut (T, U) of \overline{G} with at most δ cut edges and such that both $T \cap C$ and $U \cap C$ contain a super vertex or at least 3 regular vertices. Consider $T \cap C$. Any super vertex it contains is active with degree at least $\delta^* = (\lg n)\delta/\alpha_0$. If there are no super vertex, but three regular vertices, then by Lemma 13, there are $\delta/3$ regular vertices with a total degree at least $\delta^2/3$. Since C is trimmed, at least $\frac{2}{5}$ of the incident edges remain in C . In either case, we conclude that $\text{vol}_C(S) > s_0$, and the same holds with $S = U \cap C$, so we conclude that C is not s_0 -strong. ■

As we cut and trim components into clusters, for each component C of H , we record the smallest s for which we have certified that C is s -strong. By Lemma 19, we are done when $s \leq s_0$. To pay for the cutting and trimming, each edge is willing to pay every time it gets into a component of half the volume. Also, an edge will pay if we have certified that it was in an s -strong component, and we can now certify that it is an $s/2$ strong component. Each of these events can happen at most $\lg m$ times per an edge. An algorithm based on this amortization is presented in Sections 4.6-4.12. It will follow the same general pattern as we used for the much simpler certify-or-cut algorithm in Section 3.

4.6 Representing the components of H

As we do the cutting of H , we are generally going to store the vertex set of each component as a list sorted in order of non-decreasing degrees. This ordering is important because if we want an initial distribution spreading mass evenly on a set X of vertices, then ordering by non-decreasing degrees implies ordering by non-increasing density as required for the PageRank algorithm to be efficient.

The lists are represented by balanced binary search trees. When an edge (u, v) is deleted, the degrees of u and v are decreased, and they have to be moved in the sorted list in $O(\log n)$ time. When we cut a component, we extract the vertices of the smaller side T in $O(|T| \log n) = O(\text{vol}_H(T) \log m)$ time, regardless of the volume of the bigger side. Since a vertex can only be moved $\lg m$ times to a component of half the volume, the amortized cost is $O(\log^2 m)$ per edge for all the cutting and trimming of H .

It is not hard to improve the above amortized cost to $O(\log m)$ time per vertex, exploiting that degree changes are only by one, and that vertices have high degrees, hence that extracting a vertex has subconstant cost per incident edge, but this is not the main bottleneck for the overall performance of our algorithm.

To discover when a component is broken, we could employ a polylogarithmic dynamic connectivity algorithm [13], but actually, it is not necessary that what we perceive as components is really connected. We only view them as cut, if it is via the small side T of a low-conductance cut from Theorem 8 or Theorem 9. If one of our components is not connected, then this just implies that there is a cut with conductance 0.

4.7 Pushing from a vertex across a small cut—the issue of parallel edges

We are now going to introduce a basic technical lemma that we shall use to find low-conductance cuts. It corresponds to Lemma 12 from Section 3.1, but now we have to handle active super vertices. A new issue is that a vertex might now have many parallel edges to a few neighbors. We cannot handle this situation in general, but in our case, we will argue that it has to be a regular vertex where the parallel edges all go to super vertices, and this special structure will be critical to our solution.

Lemma 20 *Consider a trimmed component B of H , and let S be one side of a cut of B with $\leq \delta$ cut edges. Start PageRank in H placing an initial mass of 1 on a vertex v in S and push to the limit. If v is a super vertex, the mass leaving S is $o(1)$. If v is a regular vertex with a fraction ε of its edges leaving S , then the mass leaving S is $\varepsilon + o(1)$.*

Proof Suppose first that v is a super vertex. We know that v has at least $\delta^* = (\lg n)\delta/\alpha_0$ incident edges in \overline{G} , and B is trimmed, so v has at least $2\delta^*/5$ incident edges in B . The cut has at most δ edges, so the fraction of edges from v leaving S is less than $5\alpha_0/2 \lg n = o(1)$.

We now first push all the initial mass from v . The mass is spread evenly over its incident edges, so the mass escaping S is $o(1)$. Moreover, since the maximal number of parallel edges between any pair of vertices is δ , the maximal residual mass ending at any vertex is $\delta/(2\delta^*/5) = 5\alpha_0/(2 \lg n)$. The minimum degree in B is $2\delta/5$, so we end up with a maximum residual density of $25\alpha_0/(4\delta \lg n)$.

By Lemma 7, from this point forward, the net flow of mass over any edge is bounded by $(25\alpha_0/(4\delta \lg n)(2\alpha_0)) = 25/(8\delta \lg n)$, so the net flow over at most δ cut edges is bounded by $25/(8 \lg n) = o(1)$. Adding in the $o(1)$ leaving S directly from v , we get that the total mass leaving S is $o(1)$.

We now consider the case where v is not a super vertex and where the fraction ε of its incident edges leave S . As above, we first push all the mass from v , sending a fraction ε of the mass out of S . We will now study what happens with the remaining residual mass. Recall as usual that the mass from v has been distributed evenly along the edges leaving v . We now partition the residual mass, recalling from [1] that pushing mass to the limit is a linear transformation. We can therefore study what happens to different parts separately.

Consider the part r' of the residual mass that landed at neighbors of regular vertices. There are no parallel edges between regular vertices, so since the degrees are at least $2\delta/5$, they get residual mass at most $5/2\delta$ and residual density at most $25/(4\delta^2)$. By Lemma 7, the net flow from r' over any edge is therefore at most $25/(8\alpha_0\delta^2)$, so from r' we get less than $25/(8\alpha_0\delta) = o(1)$ mass leaving S over the at most δ cut edges.

For each neighbor v_i of a super vertex v , let r_i be the residual mass it receives from v . If v_i is outside S , we already count r_i as lost from S in the initial push from v , so we can assume that v_i is inside S . Our analysis above shows that when we push mass starting from a super vertex in S , then the mass leaving S is only a fraction $o(1)$, so in this case $o(r_i)$. However, $\sum_i r_i < 1$, so when we add up the limit distributions, we conclude that only $o(1)$ mass leaves S after the initial loss of ε to the neighbors of v outside S . ■

4.8 Starting from a captured vertex

Consider a vertex v in a trimmed component C . We say v is *captured* if there is a set $S \subseteq V(C)$ with $s_0 \leq \text{vol}_C(S) \leq m(C)$ and $|\partial_C(S)| \leq \delta$ that contains v and at least $\frac{3}{4}$ of the edges incident to v . If $\text{vol}_C(S) \leq s$, we further say that v is *s-captured*.

Finding a low-conductance cut is easy if we can somehow guess a captured vertex. Using Lemma 20 and Theorem 8, we will prove:

Lemma 21 *Start PageRank from a vertex v in a trimmed component C . Given $s \in [s_0, m(C)]$, we can do one of the following:*

- (i) *Find a set $A \subseteq V(C)$ with $\Phi_C(A) \leq \Phi_0$ and $\text{vol}_C(A) \leq m(C)$ in time $\tilde{O}(\text{vol}_C(A))$. If $s \leq m(C)/16$, we will have $\text{vol}_C(A) \leq 16s$ and $\text{excess}(\text{PR}_C(\{v\}), A) \geq 1/(16 \lg(4s))$.*
- (ii) *Certify in $\tilde{O}(s)$ time that v is not s -captured.*

Proof By Lemma 20, if we start PageRank with mass 1 on v that is s -captured, and push mass to the limit, we know that $3/4 - o(1)$ of the mass will stay in S by Lemma 20. Since $\text{vol}_C(S) \leq m(C)$, this corresponds to an excess of at least $3/4 - o(1) - 1/2 > 1/5$. Thus, by Theorem 8 with $G = C$ and $\gamma = 1/5$, we find a cut with small side $A = T$ and conductance $\Phi_C(A) = O(\sqrt{\alpha_0 \log m}) \ll \Phi_0$ in time $\tilde{O}(\text{vol}_C(T)/\alpha) = \tilde{O}(\text{vol}_C(A))$.

Now, if $s \leq m(C)/16$ and $\text{vol}_C(S) \leq s$, the mass $3/4 - o(1)$ corresponds to an excess of at least $3/4 - o(1) - 1/32 > 1/2$. Thus we can $\gamma = 1/2$ in Theorem 8, noting that $s \leq m(C)\gamma/8 = m(C)/16$. Then $\text{vol}_C(A) \leq 16s$, and $\text{excess}(\text{PR}_C(\{v\}), A) \geq 1/(32 \lg(4s))$. ■

We also have the following simple observation:

Observation 22 *If for some trimmed component C of H there is a set $S \subseteq V(C)$ with $4\delta \leq \text{vol}_C(S) \leq m(C)$ and $|\partial_C(S)| \leq \delta$, then S captures some vertex $v \in S$.*

Proof We have $\text{vol}_C(S) \geq 4\delta$ and $|\partial_C(S)| \leq \delta$, so $\frac{3}{4}$ of the edges with an endpoint in S stay in S , so this must also hold for at least one vertex in S . ■

4.9 Starting from set of non-captured vertices

Next we consider the case where we somehow manage to guess a large set X of vertices that are not captured.

Lemma 23 *Let $s \in [s_0, m(C)]$ and C be a trimmed component of H . Let $X \subseteq V(C)$ be a degree-ordered set of at least $64m(C)/(s\alpha_0)$ vertices that are not s -captured in C . We can then do one of the following:*

- (i) *Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$. Then, with $p^* = \text{PR}_C(\alpha_0, X)$, we have $\text{excess}_C(p^*, A) \geq 1/(128 \lg(8m))$.*
- (ii) *Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(m(C))$, and certify for the large side $B = V(C) \setminus A$, that every set S in C with $|\partial_C(S)| \leq \delta$ and $\text{vol}_C(S) \leq s$ has $\text{vol}_C(S \cap B) \leq s/2$.*
- (iii) *Certify in $\tilde{O}(m(C))$ time that there is no set S in C with $|\partial_C(S)| \leq \delta$ and $s/2 < \text{vol}_C(S) \leq s$.*

Above, it is only case (ii) and (iii) that depend on the assumption that no vertex in X is s -captured.

Proof We are going to start PageRank with mass 1 evenly spread on the vertices in X and then push it to the limit in C . Let $p^* = \text{PR}_C(X, \alpha_0)$ denote the limit distribution. Since C is trimmed, the minimum degree in C is $2\delta/5$.

Assuming that no vertex $v \in X$ is s -captured, we will argue that only little mass can end in a set $S \subseteq V(C)$ with $|\partial_C(S)| \leq \delta$ and $\text{vol}_C(S) \leq s$. We assume for now that such a set S exists and that $\text{vol}_C(S) > s/2$.

First we bound the number of vertices from X in S . Consider a vertex $v \in X \cap S$. Since v is not s -captured in C , it has at least $1/4$ of its edges in C leaving S , but C is trimmed, so v has degree at least $2\delta/5$ in C , so v has more than $\delta/10$ edges leaving S . But $|\partial_C(S)| \leq \delta$, so this implies $|X \cap S| \leq 10$, meaning that the total mass starting in S is at most $10/|X|$.

Also, initially, the maximal mass at any vertex is $1/|X|$, corresponding to a density of at most $5/(2\delta|X|)$, so by Lemma 7, the net flow over any edge is at most $5/(4\delta|X|\alpha_0)$, and hence the total net flow into S across $\partial_C(S)$ is therefore at most $5/(4\alpha_0|X|)$. The final mass in S is thus at most

$$(10 + 5/(4\alpha_0))/|X| < 4/(\alpha_0|X|) = s/(16m(C)).$$

For the inequality above, we used that $\alpha_0 = o(1) < 11/40$. Since $\text{vol}_C(S) > s/2$, this means that vertices $u \in S$ with limit density $p^*(u)/d(u) \leq 1/(4m(C))$ represent more than half the volume of S .

We now apply Theorem 9 with $\gamma = 1/2$. We get a set $A = T$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) = O(\sqrt{\alpha_0 \log m}) = O(1/(\log m)^2) \ll \Phi_0$. If we end in case (i) of Theorem 9, the set A is found quickly in time $\tilde{O}(\text{vol}_C(A)/(\gamma\alpha_0)) = \tilde{O}(\text{vol}_C(A))$ and then $\text{excess}_C(p^*, A) \geq \gamma/(64 \lg(8m(C))) = 1/(128 \lg(8m(C)))$ as claimed in (i) of the lemma.

If we end in case (ii) of Theorem 9, the set A is found in time $\tilde{O}(m(C)/(\gamma\alpha_0)) = \tilde{O}(m(C))$ with the guarantee that A contains all vertices with $p^*(u)/d(u) \leq 1/(4m(C))$, which implies that $\text{vol}_C(A \cap S) > \text{vol}_C(S)/2$. With $B = V(C) \setminus A$, this gives $\text{vol}_C(B \cap S) \leq \text{vol}_C(S)/2$, and this holds for any set S with $|\partial_C(S)| \leq \delta$ and $s/2 < \text{vol}_C(S) \leq s$, so (ii) of the lemma is satisfied.

If we end in case (iii) of Theorem 9, we know that there is no vertex u with $p^*(u)/d(u) \leq 1/(4m(C))$, but then we conclude that there is no set S with $|\partial_C(S)| \leq \delta$ and $s/2 < \text{vol}_C(S) \leq s$, so (iii) of the lemma is satisfied. ■

Lemma 24 *In Lemma 23, suppose the component C is on level i and $s = s_i$. Then, in case (ii), the large side B is on level $i + 1$, and in case (iii), C is on level $i + 1$.*

Proof Consider any cut of \overline{G} with at most δ cut edges, and let T be the side minimizing $\text{vol}_C(C \cap T)$, and set $S = C \cap T$. Since C is on level i , we know that $\text{vol}_C(S) \leq s_i$. Moreover, $|\partial_C(S)| \leq |\partial_{\overline{G}}(T)| \leq \delta$.

In case (iii), the algorithm certifies that we cannot have $s_i/2 < \text{vol}_C(S) \leq s_i$, so $\text{vol}_C(C \cap T) = \text{vol}_C(S) \leq s_{i+1} = s_i/2$, implying that C is on level $i + 1$.

In case (ii), the algorithm certifies that $\text{vol}_C(S \cap B) \leq s_i/2$, and then $\text{vol}_B(T \cap B) = \text{vol}_B(S \cap B) \leq s_i/2 = s_{i+1}$, implying that B is on level $i + 1$. ■

4.10 Recursing with large sides

We now have a simple recursive step given a trimmed component C that is certified s -strong for $s = \tilde{\Omega}(m(C))$. We simply pick an arbitrary vertex set $X \subseteq V(C)$ with $\lceil 64m(C)/(s\alpha_0) \rceil = \tilde{O}(1)$ vertices.

Then in parallel alternation, we run Lemma 21 on every vertex $v \in X$, and we run Lemma 23 on the set X . We terminate as soon as someone finds a set A with $\Phi_C(A) \leq \Phi_0$ corresponding to case (i) in Lemma 21 or in Lemma 23, calling this early termination, we continue until all processes have terminated.

In the early termination case, since we run only $\tilde{O}(1)$ processes in parallel, the total running time is $\tilde{O}(\text{vol}_C(A))$. This is paid by the $\text{vol}_C(A)$ edge end-points in A since they are now in a component of half the volume.

If no process reaches case (i), the total running time is just bounded by $\tilde{O}(m(C))$. We get from Lemma 21 (ii) that no vertex $v \in X$ is s -captured, which means that we can trust the certifications in case (ii) and (iii) of Lemma 23. Thus, in case (iii), by Lemma 24, we can now certify that C is only $s/2$ -strong, and $\tilde{O}(m)$ time is paid by the edge end-points in C . Likewise, in case (ii) of Lemma 23, by Lemma 24, we can now certify that B is only $s/2$ -strong, and $\tilde{O}(m)$ time is paid by $\text{vol}_C(B) \geq m$ edge end-points in B .

4.11 Recursing with small sides

We will now show how to recurse when we have a trimmed component C that is certified to be s -strong when $s \ll m(C)$.

First let us see what goes right and wrong if we try to do the same as we did with large sides in Section 4.10. The algorithm would still be correct, but now we have no good bound on the size of the set X . This means that the multiplicative slowdown from running $|X|$ process is not bounded.

It is instructive to note that if we apply Lemma 21 to all $v \in X$, then the total running time is $\tilde{O}(m(C))$, for the lemma spends $\tilde{O}(s)$ time on each of the $|X| = \lceil 64m(C)/(s\alpha_0) \rceil$ vertices. We will be OK spending this time if we for all $v \in X$ end in case (ii), certifying that v is not s -captured. However, we cannot afford to spend this much time if we for some $v \in X$ end in case (i) with no certification but a low-conductance cut around a very small side.

Our idea to circumvent the problem is to make sure that case (i) will not happen for any $v \in X$. We will exploit that case (i) implies a minimum amount of excess, both in Lemma 21 and in Lemma 23, and we want to detect this efficiently in advance. In some sense this is the most tricky part of our algorithm, and the motivation for including the corresponding excess guarantees in Theorem 8 and Theorem 9. The following two lemmas address the issue. The first lemma is about identifying a large set of vertices in a trimmed component C that are not s -captured without treating each one individually as in Lemma 21.

Lemma 25 *For $s \in [s_0, m(C)]$, let Y be a set of at most $m(C)/(512s \lg(4s))$ vertices from a trimmed component C . Then we can do one of the following:*

- (i) *Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$.*
- (ii) *Identify a subset $X \subseteq Y$, $|X| \geq |Y|/2$ in $\tilde{O}(m(C))$ time, certifying that no vertex in X is s -captured in C .*

Proof First we consider a simple algorithm that in $\tilde{O}(m(c))$ time will identify a set $X \subseteq Y$ with no s -captured vertices. This is, in itself trivial, since $X = \emptyset$ would do. However, here we apply Lemma 21 to each $v \in Y$ in $\tilde{O}(s)$ time. Some vertices will be reported to be not s -captured, and they are the ones we place in X . The total time we spend is $\tilde{O}(|Y|s) = \tilde{O}(m(C))$, so if X ends up with at least half the vertices from Y , then we are done.

Suppose now that the set X ends up with less than half the vertices from Y . For every $v \in Y \setminus X$, when running PageRank from v with Lemma 21, we found a low conductance cut where the small side T_v has $\text{vol}_C(T_v) \leq 16s$ and a limit excess above $1/(16 \lg(4s))$. This is also a lower bound for the limit mass in T_v .

Now consider what happens if we run PageRank evenly from Y , setting $p^\circ(v) = 1/|Y|$ for $v \in Y$; 0 otherwise. Recall that pushing to the limit is a linear transformation. The mass from a vertex $v \in Y \setminus X$ gets distributed such that at least a fraction $1/(16 \lg(4s))$ of it ends in T_v . Since $Y \setminus X$ is at least half of Y , we have $p^\circ(Y \setminus X) \geq 1/2$. It follows that when we push the mass from Y to the limit, we end up with mass $1/(32 \lg(4s))$ in the set $\overline{S}' = \bigcup_{v \in Y \setminus X} T_v$. Moreover $\text{vol}_C(\overline{S}') \leq 16s|Y|$. This means that \overline{S}' gets excess at least $\gamma' = 1/(32 \lg(4s)) - 16s|Y|/(2m(C))$. However, we have $|Y| \leq m(C)/(512s \lg(4s))$, and hence $\gamma' \geq 1/(64 \lg(4s))$. Thus, if we apply Theorem 8 with excess parameter γ' , then we get one of two outcomes:

- either we find a set A with $\text{vol}(A) \leq m(C)$ and $\Phi_C(A) \leq O(\sqrt{(\alpha_0 \log m)/\gamma'}) = O(1/(\log m)^{3/2}) \ll \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$,
- or in $\tilde{O}(m(C))$ time, we certify that there is no set S' with excess γ' . In this case, we apply Lemma 21 to every vertex in Y in $|Y|\tilde{O}(s) = \tilde{O}(m(C))$ time. We know that at least half the vertices from Y will be certified as not s -captured, and they are the ones included in X .

■

Another lemma of the same spirit will be used to certify that if we run PageRank from half the vertices in a given set in a trimmed component C , then we will not find a very small low-conductance set with the excess from Lemma 23 (i).

Lemma 26 *Let Y be any set of vertices from a trimmed component C . Then we can do one of the following:*

- Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$.
- Certify in $\tilde{O}(m(C))$ time that there is no subset $X \subseteq Y$, $|X| \geq |Y|/2$ and a set $A \subseteq V(C)$ and $\text{vol}_C(A) \leq m(C)/(256 \lg(8m))$ such that $p^* = \text{PR}_C(\alpha_0, X)$ has $\text{excess}_C(p^*, A) \geq 1/(128 \lg(8m))$.

Proof Let us assume that there is a subset $X \subseteq Y$, $|X| \geq |Y|/2$ and set $A \subseteq V(C)$ and $\text{vol}_C(A) \leq m(C)/(256 \lg(8m))$ such that $p_X^* = \text{PR}_C(\alpha_0, X)$ has $\text{excess}_C(p_X^*, A) \geq 1/(128 \lg(8m))$. This means that we have to end in case (i).

We will now consider the limit distribution $p_Y^* = \text{PR}_C(\alpha_0, Y)$ when we start PageRank with mass 1 evenly distributed on Y . Since $|X| \geq |Y|/2$, when we spread mass 1 evenly on Y instead of X , the vertices in X get at least half as much mass. Since pushing to the limit is a linear transformation, it follows for every vertex $v \in C$, that $p_Y^*(v) \geq p_X^*(v)/2$. In particular, we get that

$$p_Y^*(A) \geq p_X^*(A)/2 > \text{excess}_C(p_X^*, A)/2 \geq 1/(256 \lg(8m)).$$

Therefore

$$\text{excess}_C(p_Y^*, A) = p_Y^*(A) - \text{vol}_C(A)/(2m(C)) \geq 1/(512 \lg(8m)).$$

Thus, starting PageRank evenly from Y and applying Theorem 8 with $\gamma = 1/(512 \lg(8m))$, we will get a set A' for case (i) with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq O(\sqrt{(\alpha_0 \log m)/\gamma}) = O(1/(\log m)^{3/2}) \ll \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$. If no such a set is found, we terminate in $\tilde{O}(m(C)/(\alpha_0 \gamma)) = \tilde{O}(m(C))$, making the conclusion of case (ii). ■

We are now ready to prove our main theorem for recursing:

Theorem 27 Let $s \in [s_0, m(C)]$ and C be an s -strong trimmed component of H . We can then do one of the following:

- (i) Find a set A with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$.
- (ii) Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(m(C))$ certifying that the large side $B = V(C) \setminus A$ is $s/2$ -strong.
- (iii) Certifying in $\tilde{O}(m(C))$ time that C is $s/2$ -strong.

Proof First we pick the degree-ordered set Y of $\lceil 128m(C)/(s\alpha_0) \rceil$ vertices from C . This could just be an initial segment of the degree-ordered list of vertices in C . We assume for now that C has this many vertices. The other case will be handled later.

We divide Y into $256^2 \lg(4s)/\alpha_0 = \tilde{O}(1)$ segments Y_i , each with at most $m(C)/(512s \lg(4s))$ vertices. For the sake of the PageRank algorithm, we cut Y and the Y_i from the vertex list that is sorted by non-decreasing degrees. This is all done in $\tilde{O}(1)$ time.

We will then, alternating in parallel, apply Lemma 25 to every Y_i while, also in parallel, applying Lemma 26 to $Y = \bigcup_i Y_i$. If any one of these ends in case (i), then this corresponds to case (i) of the theorem. The multiplicative $\tilde{O}(1)$ slowdown does not affect the time bound.

Assume case (i) does not apply. Then for each Y_i , by Lemma 25 (ii), we find a subset $X_i \subseteq Y_i$ with at least half the vertices and such that no vertex in X_i is s -captured. Then no vertex in $X = \bigcup_i X_i$ is s -captured, and it has at least half the vertices from Y . Now by Lemma 26 (ii), we know that there is no set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)/(256 \lg(8m))$ so that for $p_X^* = \text{PR}_C(\alpha_0, X)$, we have $\text{excess}_C(p_X^*, A) \geq 1/(128 \lg(8m))$.

We have spent $\tilde{O}(m(C))$ time so far. Now we sort at least $64m(C)/(s\alpha_0)$ vertices from X by degree, and then we apply Lemma 23, but because of the limited excess guarantee, we know that if we end in case (i), then the set A cannot have volume below $m(C)/(256 \lg(8m))$, and therefore the total time we have spent to find A can be stated as $\tilde{O}(m(C)) = \tilde{O}(\text{vol}_C(A))$, as required for case (i) of the theorem.

The other cases are the same as in Lemma 23, with the conclusions from Lemma 24 added.

When there are not enough vertices Above we assumed that we could pick $\lceil 128m(C)/(s\alpha_0) \rceil$ vertices from C , but C might not have this many vertices. Then we can no longer hope to identify the set X of $64m(C)/(s\alpha_0)$ vertices needed for a direct application of Lemma 23. We will therefore need a modified strategy.

This time we pick all the vertices from C setting $Y = V(C)$. As above, we partition into $\tilde{O}(1)$ sets Y_i , each with at most $m(C)/(256s \lg(4s))$ vertices (if Y is small we get fewer sets Y_i , which is only an advantage).

We want to apply PageRank as above, but instead of spreading the initial mass evenly on Y and the Y_i , we will spread it so as to get even densities (see below). This requires slight modifications of several of our lemmas.

We want to apply Lemma 25 in parallel to every Y_i , but in Lemma 25 (ii), instead of getting a subset X_i with $|X_i| \geq |Y_i|/2$, we want $\text{vol}_C(X_i) \geq \text{vol}_C(Y_i)/2$. The proof of this is almost identical. When starting PageRank, this time we spread the density evenly on Y_i so that each vertex $v \in Y_i$ has density $p^\circ(v)/d(v) = 1/\text{vol}_C(Y_i)$. If $\text{vol}_C(X_i) \leq \text{vol}_C(Y_i)/2$, then $p^\circ(Y_i \setminus X_i) \geq 1/2$. Nothing else needs to be changed in the proof of Lemma 25. Assuming that none end in case (i), then for each Y_i , we get a subset

X_i with $\text{vol}_C(X_i) \geq \text{vol}_C(Y_i)/2$ and such that no vertex in X_i is s -captured. We consider now the set $X = \bigcup_i X_i$. It has $\text{vol}_C(X) \geq \text{vol}_C(Y)/2 = m(C)$, and there is no s -captured vertex in X .

Interestingly, we get the conclusion of Lemma 26 (ii) without having to apply its algorithm to $Y = \bigcup_i Y_i = V(C)$. More precisely, let p_X° be the even density distribution on X , that is $p_X^\circ(v)/d(v) = 1/\text{vol}_C(X)$ if $v \in X$; 0 otherwise. Then p_X° is dominated by the stationary distribution $1/(\text{vol}_C(X))$, and hence so is $p_X^* = \text{PR}_C(\alpha_0, p_X^\circ)$. This means that a set A has $\text{excess}_C(p_X^*, A) \geq \text{vol}_C(A)(1/\text{vol}_C(X) - 1/(2m(C))) \geq \text{vol}_C(A)/(2m(C))$. Thus, to have $\text{excess}_C(p_X^*, A) \geq 1/(128 \lg(8m))$, we need $\text{vol}_C(A) \geq m(C)/(64 \lg(8m)) = \tilde{\Omega}(m(C))$.

We have spent $\tilde{O}(m(C))$ time, and now we want a variant of Lemma 23 that we can apply to our even density distribution p_X° . Since it is dominated by the stationary distribution $1/(m(C))$. Therefore, by Lemma 7, the net flow over any edge is at most $1/(\alpha_0 m(C))$.

We know that no vertex $v \in X$ is s -captured. We will argue that only little mass can end in a set $S \subseteq V(C)$ with $|\partial_C(S)| \leq \delta$ and $\text{vol}_C(S) \leq s$. We assume for now that such a set S exists and that $\text{vol}_C(S) > s/2$.

A vertex $v \in S$ that is not s -captured has at least $\frac{1}{4}$ of its edges leaving S , so with $|\partial_C(S)| \leq \delta$, we conclude that the total degree of such vertices is 8δ , implying that their total initial mass is at most $8\delta/m(C)$. Including the mass the flows into S over $\partial_C(S)$, we get that the total mass ending in S is at most

$$p_X^* \leq 8\delta/m(C) + 2\delta/(\alpha_0 m(C)) < 4\delta/(\alpha_0 m(C)) \leq s/(16m(C)).$$

The second inequality uses that $\alpha_0 \leq 1/8$, and the third uses that $s \geq s_0 = 64\delta/\alpha_0$.

Having proved that $p_X^* \leq s/(16m(C))$, the rest of the proof is just like that of Lemma 23, applying Theorem 9 with $\gamma = 1/2$ to p_X° . If we end in case (i), our excess limit implies that $\text{vol}_C(A) \geq m(C)/(64 \lg(8m)) = \tilde{\Omega}(m(C))$. The other cases are the same as in Lemma 23, with the conclusions from Lemma 24 added. ■

4.12 Getting to the clusters and proving the main theorems

In our process of cutting and trimming H down to clusters, each component C is certified as being s -strong for some s . If $m(C) \leq s$, we set $s = m(C)$. Being s -strong is automatically inherited by subgraphs, but our aim is to reduce it down to s_0 where we know we have a cluster by Lemma 19. Assuming $s > s_0$, we apply Theorem 27 and get one of the following:

- (i) Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(\text{vol}_C(A))$.
- (ii) Find a set $A \subseteq V(C)$ with $\text{vol}_C(A) \leq m(C)$ and $\Phi_C(A) \leq \Phi_0$ in time $\tilde{O}(m(C))$ certifying that the large side $B = V(C) \setminus A$ is $s/2$ -strong.
- (iii) Certifying in $\tilde{O}(m(C))$ time that C is $s/2$ -strong.

In case (i), the edges incident to A pay for the $\tilde{O}(\text{vol}_C(A))$ time since they are now in a component of half the volume.

Case (iii) certifies that C is $s/2$ -strong, so the $\tilde{O}(m(C))$ time is paid by the edge end-points in C . Likewise, case (ii) certifies that $C \cap B$ is $s/2$ -strong, and the $\tilde{O}(m(C))$ time is paid by the $\text{vol}_C(B) \geq m(C)$ edge end-points in B .

An edge can only pay $\lg m$ time for getting into half the volume and $\lg m$ times for getting the strength of its component halved, so in total it pays on $\tilde{O}(1)$ to be either removed or end in a cluster.

The cluster cores are contracted, and then we restart. As described at the end of Section 4.4, the contractions halve the number of edges, so this loop is also only iterated a logarithmic number of times. Thus, in $\tilde{O}(m)$ time, we find the contracted graph \bar{G} with $\tilde{O}(m/\delta)$ edges which contains all non-trivial min-cuts of G . This completes the proof of Theorem 1, and then a min-cut of G is found in $\tilde{O}(m)$ time using Gabow's algorithm as described in Corollary 2.

We will now finally show to bootstrap Theorem 1 to prove Theorem 3 so that the contracted graph has only $\tilde{O}(n)$ edges and $\tilde{O}(n/\delta)$ vertices.

First we take out disjoint forests F_1, \dots, F_δ from G such that F_i is maximal in $G \setminus \bigcup_{j < i} F_j$. Nagamochi and Ibaraki [24] have described how to do this in linear time. Then $H = \bigcup_i F_i$ has at most $m_H = \delta n$ edges. Moreover, H preserves all cuts of size $\leq \delta$ and larger cuts preserve at least δ of their edges. Finally, no min-cut of G cuts any edge left in $G \setminus H$. We are later going to contract these remaining edges, but we cannot destroy simplicity yet.

We now apply Theorem 1 to H , obtaining a contracted graph \bar{H} with $\bar{m}_H = \tilde{O}(m_H/\delta) = \tilde{O}(n)$ edges. We claim that \bar{H} has only $\tilde{O}(n/\delta)$ vertices. Trivially \bar{H} has at most $2\bar{m}_H/\delta = \tilde{O}(n/\delta)$ vertices of degree $\geq \delta$, which was the minimum degree in G . This means that a lower degree vertex in \bar{H} is a contracted vertex set S from H forming a side of a non-trivial cut with less than δ cut edges. The same proof as that of Observation 5 shows that S has at least δ vertices, but this means that we can have at most n/δ contracted vertices with degree below δ . Thus we conclude that \bar{H} has at most $\tilde{O}(n/\delta)$ vertices.

Our final step is to contract the end-points of each edge left in $G \setminus H$. These contractions can only decrease the number of edges and vertices, and the resulting graph \bar{G} is obtained from G using contractions only. This implies that all cuts in \bar{G} are cuts in G and we know that all non-trivial min-cuts are preserved. This completes the proof of Theorem 3.

In the introduction, we also mentioned that with edge connectivity λ , it is straightforward to generalize our algorithms to preserve all non-trivial cuts with less than $(2 - \varepsilon)\lambda$ edges for a given $\varepsilon = \Omega(1)$. First we have an obvious change in the definition of a cluster. Currently a trimmed component C (with internal degrees at least $2\delta/5$) is a cluster if every cut of at most δ edges from \bar{G} has no more than 2 regular vertices from C on both sides. Now it should be a cluster if every cut of at most $(2 - \varepsilon)\delta$ edges of \bar{G} has no more than 4 regular vertices from C on both sides. Another change is the definition of loose vertices that do not belong to the core of the cluster C . Currently a vertex $v \in C$ is loose if at least $d(v)/2 - 1$ of its edges leave C . Now, to be loose, v should have at least $\varepsilon d(v)/2 - 3$ edges leaving C . Everything else just has to be changed accordingly.

4.13 Log-factors

In this paper, we have not worried about the number of log-factors in our near-linear time bound for solving the min-cut problem. We will now briefly discuss how many we need. Currently, we have $\alpha_0 = 1/(\log m)^5$, but in fact it suffices with $\alpha_0 = 1/(c_0(\log m)^4)$ for some sufficiently large constant c_0 . The place that puts the biggest demand on α_0 is in the end of the proof of Lemmas 25 and 26 where we need that $\Phi_C(A) = O(\sqrt{(\alpha_0 \log m)/\gamma}) = O(\sqrt{\alpha_0 \log^2 m}) \leq \Phi_0 = 1/(20 \lg m)$. By definition of the O -notation, there exists a large enough constant c_0 such that $\alpha_0 = c_0(\log m)^4$ yields $\Phi_C(A) \leq 1/(20 \lg m)$.

We can also reduce the requirement on δ to $\delta \geq c_1/\alpha_0$ and set $\delta^* = c_1\delta/\alpha_0$ for some sufficiently large constant c_1 . The critical place is Lemma 20 which currently says that if we start the PageRank algorithm from a vertex with a fraction ε of its edges leaving a certain set S , then in the limit, the mass leaving S is

only $\varepsilon + o(1)$. If we instead parameterize by c_1 and change the proof of Lemma 20 accordingly, the mass leaving S is at most $\varepsilon + 50/(8c_1)$. When we later apply Lemma 20 to the proof of Lemma 21, what we need is that $3/4 - 50/(8c_1) - 1/2 > 1/5$, which is true if $c_1 > 125$. We note here that the calculations are not set up to minimize constants.

The conclusion is that we can run our algorithm with parameters $\alpha_0 = O(\log^4 m)$ and $\delta^* = O(\delta \log^4 m)$. For Lemma 16, this implies that the number of edges leaving passive super vertices is $O(m\delta^*/\delta^2) = O(m(\log^4 m)/\delta)$, which then also bounds the number of edges in \overline{G} .

The bottleneck in time originates from Lemma 25 (i), where the set A is really found in time $O(\text{vol}_C(A)(\log m)/(\gamma\alpha_0)) = O(\text{vol}_C(A) \log^6 m)$ time. In the proof of Theorem 27, we run $O(\log s/\alpha_0) = O(\log^5 m)$ such experiments from Lemma 25 in parallel, so the cost is $O(\log^{11} m)$ per edge, and the same edge may get charged $\lg m$ times as it ends up in smaller sets. Thus a total cost of $O(\log^{12} m)$ per edge is needed in order to find the clusters. When we afterwards contract the cores, we halve the number of edges, so it is the cost of the first cluster finding round that counts. Our total cost for finding \overline{G} is $O(m \log^{12} m)$. Since \overline{G} has only $O(m(\log^4 m)/\delta)$ edges, using Gabow's algorithm, we can now find a minimum cut in $O(m(\log^5 m)/\delta)$ time. Our overall time bound for finding the minimum cut is thus $O(m \log^{12} m) = O(m \log^{12} n)$.

5 Limit concentration and low conductance cuts: the proofs

In this section we will prove the Theorems from Section 2.1. The analysis is self-contained but uses some of the techniques from [1].

5.1 Sweeping for low conductance cuts in linear time

We will first present a simple variant of the approximate PageRank in Algorithm 2 which makes the sweep for a low conductance cut run in linear time, even on a pointer machine. The issue is that in order to do the sweep, we need the vertices to be sorted according to the settled mass density.

First we note that we can make the push more flexible in how much residual mass we push around, as described in Algorithm 4. In the approximate PageRank in Algorithm 2, we pushed a vertex u if it had

Algorithm 4: Push'(α, u, q)—assumes $r(u) \geq q$

$p(u) \leftarrow p(u) + \alpha q;$

for $(u, v) \in E$ **do** $r(v) \leftarrow r(v) + (1 - \alpha)q/(2d(u)); r(u) \leftarrow r(u) - (1 + \alpha)q/2.$

$r(u)/d(u) \geq \varepsilon \iff r(u) \geq \varepsilon d(u)$, but now we will only push $\varepsilon d(u)$ of the residual mass. Thus we get the revised approximate PageRank in Algorithm 5. The push at u settles mass exactly $\varepsilon \alpha d(u)$ in $p(u)$.

Algorithm 5: ApproximatePageRank'($\alpha, \varepsilon, p^\circ$)

$r \leftarrow p^\circ; p \leftarrow 0^V;$

while $\exists u : r(u)/d(u) \geq \varepsilon$ **do** Push'($\alpha, u, \varepsilon d(u)$)

As noted in [1], since the initial mass is 1, the sum of the degrees of the pushes is bounded by $1/(\varepsilon\alpha)$. As described in [1], we can implement approximate PageRank in constant time per edge incident to a vertex pushed, and hence in $O(1/(\varepsilon\alpha))$ total time.

Using our Algorithm 5 for approximate PageRank, when we push at u , the settled density $p(u)/d(u)$ grows by exactly $\varepsilon\alpha$. We now maintain a list of groups $i = 0, 1, \dots$, where group i is a doubly linked list of vertices with settled density $i\varepsilon\alpha$. Each vertex also has a pointer to the head of its group, and the head has a pointer to the head of the next group. Thus, when we push u , we just have to pull it out of its group, and follow two pointers to get to the head of the next group where it is inserted. With this structure, we have direct access to vertices in order of non-increasing settled density, and it is produced on the fly as we run our approximate PageRank in $O(1/(\varepsilon\alpha))$ total time.

5.2 High densities

We will now study vertices with densities above t_0 where

$$\text{vol}(V_{\geq t_0}^p) \leq m. \quad (2)$$

By definition, for any $t > t_0$, we have conductance $\Phi(V_{\geq t}^p) = |\partial(V_{\geq t}^p)| / \text{vol}(V_{\geq t}^p)$. Assuming (2), for any $\tau \in (t_0, 1]$, we will prove

$$\min_{t \in (t_0, \tau]} \Phi(V_{\geq t}^p) \leq \sqrt{\frac{42\alpha}{(\tau - t_0)\text{vol}(V_{\geq \tau}^p)}}. \quad (3)$$

In fact, for any given $\phi \leq \min_{t \in (t_0, \tau]} \Phi(V_{\geq t}^p)$, we are going to prove (3) in the following equivalent form.

Lemma 28 $\tau - t_0 \leq \frac{42\alpha}{\phi^2 \text{vol}(V_{\geq \tau}^p)}$.

Let $t \in (t_0, \tau]$. By (2), we have $\text{vol}(V_{\geq t}^p) \leq m$, so by definition, $|\partial(V_{\geq t}^p)| \geq \phi \text{vol}(V_{\geq t}^p)$. Consider any edge (u, v) leaving $V_{\geq t}^p$ (so $u \in V_{\geq t}^p$ but $v \notin V_{\geq t}^p$). By Fact 6 (and since $\alpha < 1/3$), the net flow over this edge from u to v is at least $(p(u)/d(u) - p(v)/d(v))/(3\alpha)$. Since $p(u)/d(u) \geq t > p(v)/d(v)$ this flow is always positive away from $V_{\geq t}^p$. Let t' be the median density $p(v)/d(v)$ of a neighbor of $V_{\geq t}^p$, counting $p(v)/d(v)$ with the multiplicity of the number of edges from $V_{\geq t}^p$ to v . We then have at least $|\partial(V_{\geq t}^p)|/2$ edges from $V_{\geq t}^p$ to vertices v with $p(v)/d(v) \leq t'$, so the net flow out of $V_{\geq t}^p$ is at least

$$\left(|\partial(V_{\geq t}^p)|/2\right)(t - t')/(3\alpha) \geq \phi \text{vol}(V_{\geq t}^p)(t - t')/(6\alpha). \quad (4)$$

But this can be no more than the total mass, which is 1, so

$$(t - t') \leq \frac{6\alpha}{\phi \text{vol}(V_{\geq t}^p)}. \quad (5)$$

For the next reasoning, we will work with the internal volume of a set $S \subseteq V$ defined as

$$\text{int-vol}(S) = 2|E \cap E(S)| = \text{vol}(S) - |\partial(S)|.$$

By definition of t' , we have the following inequalities:

$$\text{int-vol}(V_{\geq t'}^p) \geq \text{vol}(V_{\geq t'}^p) - |\partial(V_{\geq t'}^p)|/2 \geq \text{vol}(V_{\geq t'}^p)/2. \quad (6)$$

$$\text{int-vol}(V_{\geq t'}^p) \geq \text{int-vol}(V_{\geq t}^p) + |\partial(V_{\geq t}^p)|/2 \geq (1 + \phi/2)\text{int-vol}(V_{\geq t}^p), \quad (7)$$

Inductively, we claim for any $t \leq \tau$ that

$$t - t_0 \leq \frac{18\alpha}{\phi^2 \text{int-vol}(V_{\geq t}^p)}.$$

Note that if $t \leq t_0$, the statement is trivially true. Applying (5), (7), and the inductive hypothesis to $t' < t$, we get

$$\begin{aligned} t - t_0 &\leq (t - t') + (t' - t_0) \\ &\leq \frac{6\alpha}{\phi \text{vol}(V_{\geq t}^p)} + \frac{18\alpha}{\phi^2 \text{int-vol}(V_{\geq t'}^p)} \\ &\leq \frac{6\alpha}{\phi \text{int-vol}(V_{\geq t}^p)} + \frac{18\alpha}{\phi^2 (1 + \phi/2) \text{int-vol}(V_{\geq t}^p)} \\ &\leq \frac{18\alpha}{\phi^2 \text{int-vol}(V_{\geq t}^p)} (\phi/3 + 1/(1 + \phi/2)) \\ &\leq \frac{18\alpha}{\phi^2 \text{int-vol}(V_{\geq t}^p)}. \end{aligned}$$

Invoking (6) and defining τ' from τ as t' from t , we get

$$\begin{aligned} \tau - t_0 &\leq (\tau - \tau') + (\tau' - t_0) \\ &\leq \frac{6\alpha}{\phi \text{vol}(V_{\geq \tau}^p)} + \frac{18\alpha}{\phi^2 \text{int-vol}(V_{\geq \tau'}^p)} \\ &\leq \frac{6\alpha}{\phi \text{vol}(V_{\geq \tau}^p)} + \frac{36\alpha}{\phi^2 \text{vol}(V_{\geq \tau}^p)} \\ &\leq \frac{42\alpha}{\phi^2 \text{vol}(V_{\geq \tau}^p)}. \end{aligned}$$

This completes the proof of Lemma (28), and hence of (3).

5.3 Low densities

We will now make a symmetric study of vertices below some t_0 such that

$$\text{vol}(V_{< t_0}^p) \leq m. \tag{8}$$

Note that if (2) is false then (8) is true, so for any value of t_0 , the analysis below applies if the analysis from the previous section did not apply. Now, for any $t < t_0$, we have conductance $\Phi(V_{< t}^p) = |\partial(V_{< t}^p)| / \text{vol}(V_{< t}^p)$. Symmetric to the results from Section 5.2, we will show

$$\min_{t \in [\tau, t_0]} \Phi(V_{< t}^p) \leq \sqrt{\frac{42\alpha}{(t_0 - \tau) \text{vol}(V_{< \tau}^p)}}. \tag{9}$$

For any given $\phi \leq \min_{t \in [\tau, t_0]} \Phi(V_{< t}^p)$, we are going to prove (9) in the following equivalent form.

Lemma 29 $t_0 - \tau \leq \frac{42\alpha}{\phi^2 \text{vol}(V_{<\tau}^p)}$.

Consider any $t \in [\tau, t_0)$. By (2), we have $\text{vol}(V_{<t}^p) \leq m$, so by definition, $|\partial(V_{<t}^p)| \geq \phi \text{vol}(V_{<t}^p)$. Consider any edge (u, v) leaving $V_{<t}^p$ (so $u \in V_{<t}^p$ but $v \notin V_{<t}^p$). By Fact 6 (and since $\alpha < 1/3$), the net flow over this edge from v to u is at least $(p(v)/d(v) - p(u)/d(u))/(3\alpha)$. Since $p(u)/d(u) \leq t < p(v)/d(v)$ this flow is always positive into $V_{<t}^p$. Let t' be the median density $p(v)/d(v)$ of a neighbor of $V_{<t}^p$, counting $p(v)/d(v)$ with the multiplicity of the number of edges from $V_{<t}^p$ to v . We then have at least $|\partial(V_{<t}^p)|/2$ edges from $V_{<t}^p$ to vertices v with $p(v)/d(v) \geq t'$, so the net flow into $V_{<t}^p$ is at least

$$(|\partial(V_{<t}^p)|/2)(t' - t)/(3\alpha) \geq \phi \text{vol}(V_{<t}^p)(t' - t)/(6\alpha). \quad (10)$$

But this can be no more than the total mass, which is 1, so symmetric to (5), we get

$$(t' - t) \leq \frac{6\alpha}{\phi \text{vol}(V_{<t}^p)}. \quad (11)$$

Also, since t' was the median neighboring density, corresponding to (12) and (7), we get

$$\text{int-vol}(V_{<t'}^p) \geq \text{vol}(V_{<t}^p) - |\partial(V_{<t}^p)|/2 \geq \text{vol}(V_{<t}^p)/2. \quad (12)$$

$$\text{int-vol}(V_{<t'}^p) \geq \text{int-vol}(V_{<t}^p) + |\partial(V_{<t}^p)|/2 \geq (1 + \phi/2)\text{int-vol}(V_{<t}^p), \quad (13)$$

The rest of the argument for Lemma (29) and (9) is exactly the same as the argument for Lemma (28) and (3).

5.4 A single low density

In this section we will show that just a single vertex with low density makes a big difference if we have a good bound $\varepsilon \leq 1/(2m)$ on the residual densities $r(v)/d(v)$ for every vertex v .

We are continuing from our analysis in Section 5.3 with some t_0 satisfying $\text{vol}(V_{<t_0}^p) \leq m$. Assume that there is at least one vertex u with density $p(u)/d(u) < \tau$. We will prove that

$$\min_{t \in [\tau, t_0)} \Phi(V_{<t}^p) \leq \sqrt{\frac{12(t_0 + \varepsilon)\alpha \lg m}{t_0 - \tau}}. \quad (14)$$

Let $\phi \leq \min_{t \in [\tau, t_0)} \Phi(V_{<t}^p)$. We shall reuse a lot of the analysis from Section 5.3 based on some $t \leq t_0$ and the median neighboring density t' . In Section 5.3, symmetric to the high density case, we said that the total flow into $V_{<t}^p$ is at most 1. However, here we assumed that the residual density on every vertex is bounded by ε , and then the total mass on $V_{<t}^p$ is at most $(t + \varepsilon)\text{vol}(V_{<t}^p)$. This gives us a different bound on the net flow into $V_{<t}^p$, which by (10) is at least $\phi \text{vol}(V_{<t}^p)(t' - t)/(6\alpha)$. Thus, as an alternative to (11), we have

$$\phi \text{vol}(V_{<t}^p)(t' - t)/(6\alpha) \leq (t + \varepsilon)\text{vol}(V_{<t}^p) \iff (t' - t) \leq 6(t + \varepsilon)\alpha/\phi \leq 6(t_0 + \varepsilon)\alpha/\phi. \quad (15)$$

Starting from $t = \tau'$, we consider how many times we can do the median expansion from t to t' before reaching or passing t_0 . First time we do it, we get at least one internal edge, so

$$\text{vol}(V_{\leq \tau'}^p) \geq 2.$$

In all subsequent iterations, we know from (13) that the volume grows by at least a factor $(1 + \phi/2)$, and by definition, $\text{vol}(V_{<t_0}^p) \leq m$, so we can have at most

$$\log_{(1+\phi/2)} m \leq (2/\phi) \lg m$$

iterations before we reach t_0 . Therefore

$$t_0 - \tau \leq (2/\phi)(\lg m)6(t_0 + \varepsilon)\alpha/\phi = 12(t_0 + \varepsilon)\alpha(\lg m)/\phi^2.$$

Thus we have

$$\phi \leq \sqrt{12(t_0 + \varepsilon)\alpha(\lg m)/(t_0 - \tau)}.$$

This also holds for $\phi = \min_{t \in [\tau, t_0]} \Phi(V_{<t}^p)$, so this completes the proof of (14).

5.5 Exploiting concentration

Our goal in this subsection is to provide an algorithm performing as stated in Theorem 8. For convenience, let us state here again.

Let $p^ = \text{PR}(\alpha, p^\circ)$ where $p^*(V) = p^\circ(V) = 1$. If there is a set S such that $\text{excess}(p^*, S) \geq \gamma$, then we can find a set T with $\text{vol}(T) \leq m$ and conductance*

$$\Phi(T) = O(\sqrt{(\alpha \log m)/\gamma}).$$

in time $O(\min\{m, \text{vol}(T)(\log m)\}/(\gamma\alpha))$. If no such set S exists, we can report this in $O(m/(\gamma\alpha))$ time.

Given a bound $s \leq m\gamma/8$ on $\text{vol}(S)$, we find T in time $O(\min\{s, \text{vol}(T)(\log m)\}/(\gamma\alpha))$ with the additional guarantees that $\text{vol}(T) \leq 8s/\gamma$ and $\text{excess}(p^, T) \geq \gamma/(16 \lg(4s))$, or report in $O(s/(\gamma\alpha))$ time that there is no set S with $\text{vol}(S) \leq s$ and $\text{excess}(p^*, S) \geq \gamma$.*

With a size bound We will first address the case where we have a bound $s \leq m\gamma/8$ on $\text{vol}(S)$. In this case, we will apply Algorithm 6 below. We know from Section 5.1 that it takes $O(1/(\varepsilon\alpha))$ time

Algorithm 6: BoundedNibble($\alpha, p^\circ, \gamma, s$)—assumes $s \leq \gamma m/8$

$\varepsilon \leftarrow \gamma/2$;

repeat

$\varepsilon \leftarrow \varepsilon/2$;

$p \leftarrow \text{ApproximatePageRank}(\alpha, \varepsilon, p^\circ)$;

if $\text{vol}(V_{\geq 1/(2m)+\varepsilon}^p) \geq \gamma/(8\varepsilon \lg(4s))$ **then**

return $T = V_{\geq t}^p$ where $t \in (1/(2m) + \varepsilon/2, 1/(2m) + \varepsilon]$ minimizes $\Phi(V_{\geq t}^p)$.

until $\varepsilon < \gamma/(4s)$ // ERROR;

return “There is no set S with $\text{excess}(p^*, S) \geq \gamma$ and $\text{vol}(S) \leq s$.”

to run an iteration including a possible sweep for low conductance cuts. Therefore the last iteration will dominate the total running time. In particular, if we error with $\varepsilon \in [\gamma/(8s), \gamma/(4s))$, the total time is at most $O(1/((\gamma/(8s)\alpha)) = O(s/(\gamma\alpha))$.

Suppose now that the algorithm terminates satisfying the condition of the if-statement for some $\varepsilon \geq \gamma/(4s)$. First we claim that

$$\text{vol}\left(V_{\geq 1/(2m)+\varepsilon/2}^p\right) < 8s/\gamma \leq m. \quad (16)$$

This follows because the total settled mass p is at most 1, so $\text{vol}\left(V_{\geq 1/(2m)+\varepsilon/2}^p\right) (1/(2m) + \varepsilon/2) \leq 1$ (note that $p(u)/d(u) \geq 1/(2m) + \varepsilon/2$ for $u \in V_{\geq 1/(2m)+\varepsilon/2}^p$). Therefore $\text{vol}\left(V_{\geq 1/(2m)+\varepsilon/2}^p\right) < 2/\varepsilon \leq 8s/\gamma$, as stated in (16). In particular, this implies that the returned set $T = V_{\geq t}^p \subseteq V_{\geq 1/(2m)+\varepsilon/2}^p$ has $\text{vol}(T) \leq 8s/\gamma \leq m$.

The condition of the if-statement implies that $\text{vol}(T) \geq \gamma/(8\varepsilon \lg(4s))$, and our running time is $O(1/(\varepsilon\alpha))$, which can then also be expressed as $O(\text{vol}(T)(\log m)/(\gamma\alpha))$ by removing ε . Also we get

$$\text{excess}(p^*, T) \geq \text{vol}(T)(1/(2m) + \varepsilon/2) - \text{vol}(T)/2m = (\varepsilon/2)\text{vol}(T) \geq \gamma/(16 \lg(4s)).$$

Finally we need to argue about the conductance. With $t_0^+ = 1/(2m) + \varepsilon/2$, we have (2) satisfied by (16). With $\tau^+ = 1/(2m) + \varepsilon$, it follows from (3) that

$$\begin{aligned} \min_{t \in (t_0^+, \tau^+]} \Phi(V_{\geq t}^p) &\leq \sqrt{\frac{42\alpha}{(\tau^+ - t_0^+)\text{vol}(V_{\geq \tau^+}^p)}} \leq \sqrt{\frac{84\alpha}{\varepsilon \text{vol}(V_{> 1/(2m)+\varepsilon}^p)}} \\ &\leq \sqrt{\frac{84\alpha}{\gamma/(8 \lg(4s))}} \leq O\left(\sqrt{\frac{\alpha \log m}{\gamma}}\right). \end{aligned} \quad (17)$$

This completes the proof of Theorem 8 assuming the algorithm terminates satisfying the condition of the if-statement for some $\varepsilon \geq \gamma/(4s)$.

We need to prove that this happens if there is a set S with $\text{excess}(p^*, S) \geq \gamma$ and $\text{vol}(S) \leq s$. A vertex $u \in S$ contributes $d(u) \max\{0, p^*(u)/d(u) - 1/(2m)\}$ to $\text{excess}(p^*, S)$, so vertices u with $p^*(u)/d(u) \leq 1/(2m) + \gamma/(2s)$ contribute less than $\gamma/2$. Let

$$S_1 = \{u \in S \mid p^*(u)/d(u) > 1/(2m) + \gamma/2\}$$

and for $i = 2, \dots, \lceil \lg(2s) \rceil$, define

$$S_i = \{u \in S \mid 1/(2m) + \gamma 2^{1-i} < p^*(u)/d(u) \leq 1/(2m) + \gamma 2^{1-i}\}$$

Then

$$\gamma < 2 \sum_{i=1}^{\lceil \lg(2s) \rceil} (p^*(S_i) - \text{vol}(S_i)/(2m)).$$

Thus, for some $i = \{1, \dots, \lceil \lg(2s) \rceil\}$, we have

$$p^*(S_i) - \text{vol}(S_i)/(2m) > \gamma/(2 \lg(4s)).$$

If $i > 1$ then

$$p^*(S_i) - \text{vol}(S_i)/(2m) \leq \gamma 2^{1-i} \text{vol}(S_i) \leq \gamma 2^{1-i} \text{vol}\left(V_{> 1/(2m)+\gamma 2^{1-i}}^{p^*}\right).$$

So

$$\text{vol}\left(V_{> 1/(2m)+\gamma 2^{1-i}}^{p^*}\right) > 2^{i-2}/\lg(4s). \quad (18)$$

This equation is also satisfied if $i = 1$, for then $S_1 \neq \emptyset$, so $\text{vol}(V_{>1/(2m)+\gamma 2^{-1}}^{p^*}) \geq 1$.

Now, consider the iteration of Algorithm 6 using the residual density bound $\varepsilon = \gamma 2^{-i-1}$, yielding a settled distribution p with $p^*(u)/d(u) - \varepsilon \leq p(u)/d(u) \leq p^*(u)/d(u)$ for all vertices u . Since $i \leq \lceil \lg(2s) \rceil$, we have $\varepsilon \geq \gamma/(4s)$, so we will indeed get to this value of ε unless we have terminated earlier. Then

$$V_{>1/(2m)+\varepsilon}^p \supseteq V_{>1/(2m)+\gamma 2^{-i}}^{p^*}$$

so

$$\text{vol}(V_{>1/(2m)+\varepsilon}^p) \geq \text{vol}(V_{>1/(2m)+\gamma 2^{-i}}^{p^*}) \geq 2^{i-2}/\lg(4s) = \gamma/(8\varepsilon \lg(4s)).$$

Indeed this means that the condition of the if-statement in Algorithm 6 is satisfied. This completes the proof of Theorem 8 when a size bound s is given.

Without a size bound With no size bound s on $\text{vol}(S)$, we will run Algorithm 7 below, claiming that it satisfies that the statement of Theorem 8. Algorithm 7 has a lot of similarities with Algorithm 6 applied

Algorithm 7: Nibble(α, p°, γ)

```

 $\varepsilon \leftarrow \gamma/2;$ 
repeat
     $\varepsilon \leftarrow \varepsilon/2;$ 
     $p \leftarrow \text{ApproximatePageRank}(\alpha, \varepsilon, p^\circ);$ 
    if  $\text{vol}(V_{\geq 1/(2m)+\varepsilon/2}^p) \leq m$  and  $\text{vol}(V_{\geq 1/(2m)+\varepsilon}^p) \geq \gamma/(8\varepsilon \lg(8m))$  then
        return  $T = V_{\geq t}^p$  where  $t \in (1/(2m) + \varepsilon/2, 1/(2m) + \varepsilon]$  minimizes  $\Phi(V_{\geq t}^p)$ .
    if  $\text{vol}(V_{\leq 1/(2m)-\varepsilon}^p) \leq m$  and  $\text{vol}(V_{\leq 1/(2m)-2\varepsilon}^p) \geq \gamma/(8\varepsilon \lg(8m))$  then
        return  $T = V_{\leq t}^p$  where  $t \in [1/(2m) - 2\varepsilon, 1/(2m) - \varepsilon]$  minimizes  $\Phi(V_{\leq t}^p)$ .
until  $\varepsilon < \gamma/(8m)$  // ERROR;
return "There is no set  $S$  with  $\text{excess}(p^*, S) \geq \gamma$ ."

```

with the trivial volume bound $s = 2m$, but instead of always returning a set T of high density vertices, it may also return a set of low density vertices. The first condition in each if-statement ensures that the set T returned has $\text{vol}(T) \leq m$. The running time analysis is exactly the same as that for Algorithm 6 with $s = 2m$.

Assume now that

$$\text{vol}(V_{\geq 1/(2m)}^{p^*}) \leq m \tag{19}$$

Then we always have

$$\text{vol}(V_{\geq 1/(2m)+\varepsilon/2}^p) \leq \text{vol}(V_{\geq 1/(2m)}^p) \leq \text{vol}(V_{\geq 1/(2m)}^{p^*}) \leq m \tag{20}$$

In particular this means that we always satisfy the first condition of the first if-statement, so Algorithm 7 behaves exactly the same as Algorithm 6 with $s = 2m$. This might violate the condition $s \leq \gamma m/8$, but in the analysis of Algorithm 6, the condition $s \leq \gamma m/8$ was only used to argue that $\text{vol}(V_{\geq 1/(2m)+\varepsilon/2}^p) \leq m$ in (16), but now this is tested directly by the algorithm. Our bounds for Algorithm 6 therefore also hold for Algorithm 7 with $s = 2m$. In particular, we get the low conductance from (17), as required for Theorem 8.

Cutting low densities To complete the analysis of Algorithm 7, we consider the case where $\text{vol}(V_{<1/(2m)}^{p^*}) < m$, and hence

$$\text{vol}(V_{<1/(2m)-\varepsilon}^p) \leq \text{vol}(V_{<1/(2m)}^{p^*}) < m \quad (21)$$

is always satisfied, while the first condition of the first if-statement is never satisfied.

Since $p^*(V) = 1$ and $\text{vol}(V) = 2m$, we have

$$\begin{aligned} \text{vol}(V_{<1/(2m)}^{p^*}) / (2m) - p^*(V_{<1/(2m)}^{p^*}) &= p^*(V_{\geq 1/(2m)}^{p^*}) - \text{vol}(V_{\geq 1/(2m)}^{p^*}) / (2m) \\ &\geq \text{vol}(S) / (2m) - p^*(S) \geq \gamma. \end{aligned}$$

We can thus make an analysis of p^* for densities below $1/(2m)$ which is symmetric to the one we did with densities above $1/(2m)$ but based on $V_{\leq 1/(2m)}^p$ instead of S . Corresponding to (18), we find an $i \leq \lceil \lg(2s) \rceil$ such that

$$\text{vol}(V_{<1/(2m)-\gamma 2^{-i}}^{p^*}) \geq 2^{i-2} / \lg(4s). \quad (22)$$

Since p^* is non-negative, we must have $i \geq \lg(2\gamma m)$, but we will not exploit this in the analysis. Now, consider the run of Algorithm 7 using the residual density bound $\varepsilon = \gamma 2^{-i-1}$. Since p^* dominates p , we get

$$V_{<1/(2m)-2\varepsilon}^p = V_{<1/(2m)-\gamma 2^{-i}}^p \supseteq V_{<1/(2m)-\gamma 2^{-i}}^{p^*}$$

so

$$\text{vol}(V_{<1/(2m)-2\varepsilon}^p) \geq 2^{i-2} / \lg(4s) = \gamma / (8\varepsilon \lg(4s)). \quad (23)$$

Thus, assuming $\text{vol}(V_{<1/(2m)}^{p^*}) \leq m$, and hence (21), we conclude that unless we stop earlier, the loop ends up satisfying the condition for returning with the second if-statement with a settled distribution p and a residual density bound ε satisfying (23).

Concerning the conductance, with $t_0^- = 1/(2m) - \varepsilon$, we have (8) satisfied by (21). With $\tau^- = 1/(2m) - 2\varepsilon$, it follows from (9) and (23), that

$$\begin{aligned} \min_{t \in [\tau^-, t_0^-]} \Phi(V_{<t}^p) &\leq \sqrt{\frac{42\alpha}{(t_0^- - \tau^-) \text{vol}(V_{<\tau^-}^p)}} \leq \sqrt{\frac{42\alpha}{\varepsilon \text{vol}(V_{<1/(2m)-2\varepsilon}^p)}} \\ &\leq \sqrt{\frac{42\alpha}{\gamma / (8 \lg(4s))}} \leq O\left(\sqrt{\frac{\alpha \log m}{\gamma}}\right), \end{aligned}$$

This completes the proof of Theorem 8.

5.6 Exploiting single low density

In this subsection we present Algorithm 8, proving that it performs as stated in Theorem 9:

Let $p^ = pr(\alpha, p^\circ)$ where $p^*(V) = p^\circ(V) = 1$. If there is a vertex u with*

$$p^*(u)/d(u) \leq (1 - \gamma)/(2m),$$

then we can find a set T with $\text{vol}(T) \leq m$ and conductance

$$\Phi(T) = O(\sqrt{(\alpha \log m)/\gamma}).$$

in time $O(m/(\gamma\alpha))$. In fact, we will obtain one of following cases:

- (i) $\text{excess}(p^*, T) \geq \gamma/(64 \lg(8m))$ and T is found in time $O(\min\{m, \text{vol}(T)(\log m)\}/(\gamma\alpha))$.
- (ii) T contains all small density vertices u with $p^*(u)/d(u) = (1 - \gamma)/(2m)$.
- (iii) A certification that there is no vertex u with $p^*(u)/d(u) = (1 - \gamma)/(2m)$.

We will not decide which case we get, but we will know which case we got.

Algorithm 8: SomeSmall(α, p°, γ)

```

 $\gamma' = \gamma/4;$ 
 $\varepsilon \leftarrow \gamma'/2;$ 
repeat
   $\varepsilon \leftarrow \varepsilon/2;$ 
   $p \leftarrow \text{ApproximatePageRank}(\alpha, \varepsilon, p^\circ);$ 
  if  $\text{vol}\left(V_{\geq 1/(2m) + \varepsilon/2}^p\right) \leq m$  and  $\text{vol}\left(V_{\geq 1/(2m) + \varepsilon}^p\right) \geq \gamma'/(8\varepsilon \lg(8m))$  then
    return Case (i):  $T = V_{\geq t}^p$  where  $t \in (1/(2m) + \varepsilon/2, 1/(2m) + \varepsilon]$  minimizes  $\Phi(V_{\geq t}^p)$ .
  until  $\varepsilon < \gamma'/(8m);$ 
 $\varepsilon \leftarrow \gamma/(8m);$ 
 $p \leftarrow \text{ApproximatePageRank}(\alpha, \varepsilon, p^\circ);$ 
if  $\exists u : p(u)/d(u) \geq (1 - \gamma)/(2m)$  then
  return Case (ii):  $T = V_{< t}^p$  where  $t \in ((1 - 0.75\gamma)/(2m), (1 - \gamma)/(2m)]$  minimizes  $\Phi(V_{< t}^p)$ .
else
  return Case (iii): “There is no vertex  $u$  with  $p^*(u)/d(u) \leq (1 - \gamma)/(2m)$ .”

```

Assume first that

$$\text{vol}\left(V_{< (1-\gamma/2)/(2m)}^{p^*}\right) > m. \quad (24)$$

If so, we have negative concentration

$$\text{vol}\left(V_{< (1-\gamma/2)/(2m)}^{p^*}\right)/(2m) - p^*\left(V_{< (1-\gamma/2)/(2m)}^{p^*}\right) > m\gamma/(4m) = \gamma/4 = \gamma'.$$

Then $S = V_{> 1/(2m)}^{p^*}$ has

$$\text{excess}(p^*, S) = \text{vol}\left(V_{\geq 1/(2m)}^{p^*}\right)/(2m) - p^*\left(V_{\geq 1/(2m)}^{p^*}\right) = p^*\left(V_{< 1/(2m)}^{p^*}\right) - \text{vol}\left(V_{< 1/(2m)}^{p^*}\right)/(2m) > \gamma'.$$

From (24) we also get $\text{vol}\left(V_{\geq 1/(2m)}^{p^*}\right) \leq m$ so (20) is satisfied. This means that the first part of Algorithm 8 behaves exactly as Algorithm 7 which with (2) can only apply the first if-statement. As we saw before, the conditions of the if-statement imply that $\text{excess}(p^*, T) \geq (\varepsilon/2)\text{vol}(T) \geq \gamma'/(16 \lg(8m)) = \gamma/(64 \lg(8m))$ and that $\Phi(T) = O\left(\sqrt{\frac{\alpha \log m}{\gamma}}\right)$. This completes the proof of case (i) of Theorem 9.

If the first part of Algorithm 8 fails to find a T as above, then we know that (24) is false, hence

$$\text{vol}\left(V_{\geq (1-\gamma/2)/(2m)}^{p^*}\right) \geq m.$$

As in Algorithm 8, we set $\varepsilon = \gamma/(8m)$ and $p \leftarrow \text{ApproximatePageRank}(\alpha, \varepsilon, p^\circ)$. With $t_0 = (1 - 0.75\gamma)/(2m)$, we get

$$\text{vol}(V_{<t_0}^p) \leq m. \quad (25)$$

Set $\tau = (1 - \gamma)/(2m)$. Since p^* dominates p , if $p^*(u)/d(u) < \tau$, then $p(u)/d(u) < \tau$. Thus, if there is no u with $p(u)/d(u) \leq \tau$, then we conclude that there is no u with $p^*(u)/d(u) < \tau$, which is our error case (iii). Otherwise we get from (14) that

$$\min_{t \in [\tau, t_0]} \Phi(V_{<t}^p) \leq \sqrt{\frac{12(t_0 + \varepsilon)\alpha \lg m}{t_0 - \tau}} = \sqrt{\frac{12((1 - 0.5\gamma)/(2m))\alpha \lg m}{\gamma/(8m)}} = O\left(\sqrt{\frac{\alpha \lg m}{\gamma}}\right)$$

The returned set $T = V_{<t}^p$ has $\text{vol}(T) \leq \text{vol}(V_{<t_0}^p) \leq m$ and it contains $V_{<\tau}^p$ including every u with $p^*(u)/d(u) < \tau$, as required for Case (ii).

The smallest ε encountered is $\varepsilon = \gamma/(8m)$, so the total running time is bounded by $O(1/(\varepsilon\alpha)) = O(m/(\gamma\alpha))$. This completes the proof of Theorem 9.

6 Cactus

Recall that the set $\partial(U)$ of edges connecting U and $T = V \setminus U$ is called a *cut* while U and T are the *sides* of the cut.

We call a loopless and 2-edge-connected graph G a *cactus* if each edge belongs to exactly one cycle. This is equivalent to saying that all blocks are cycles (allowing two-element cycles). For example, a cactus may be obtained by duplicating each edge of a tree. Note that the minimum cuts of a cactus C are exactly those pairs of edges which belong to the same cycles of C .

The following result states that the minimum cuts of an arbitrary graph have the same structure as the minimum cuts of a cactus.

Theorem 30 (Dinitz, Karzanov, and Lomonosov, [4]) *Let λ be an integer and $G = (V, E)$ a loopless graph for which the cardinality of a minimum cut is λ . There is a cactus $C = (U, F)$ and a mapping ϕ from V to U so that the pre-images $\phi^{-1}(U_1)$ and $\phi^{-1}(U_2)$ are the two sides of a minimum cut of G for every 2-element cut of C with sides U_1 and U_2 . Moreover, every minimum cut of G arises this way.*

Gabow's cactus algorithm [8] can construct the cactus and mapping of Theorem 30 in $\tilde{O}(\lambda m)$ time, but here we will do the construction in near-linear time if the input graph is simple.

Theorem 31 *There is a near-linear time algorithm that given a simple graph $G = (V, E)$ constructs a cactus $C = (U, F)$ and a mapping ϕ from V to U as described in Theorem 30.*

Proof Let δ be the minimum degree in G . First we apply our main technical result, Theorem 1, which contracts vertex sets in near-linear time, producing a graph $\overline{G} = (\overline{V}, \overline{E})$ with $\overline{m} = \tilde{O}(m/\delta)$ edges such that all non-trivial min-cuts of G are preserved in \overline{G} .

As in the proof of Corollary 2, we now run Gabow's edge-connectivity algorithm [9] on \overline{G} , asking it to fail if the edge-connectivity $\overline{\lambda}$ of \overline{G} is above δ . This takes $\tilde{O}(\delta \overline{m}) = \tilde{O}(m)$ time, and now we compare $\overline{\lambda}$ with the min-degree δ .

If $\overline{\lambda} > \delta$, then all min-cuts in G are trivial and then the edge connectivity λ of G is the minimum degree δ . Let v_1, \dots, v_h denote the vertices of degree δ . Let $U = \{u_0, u_1, \dots, u_h\}$ be the vertex set of a cactus

$C = (U, F)$ in which u_0 and u_i are connected by two parallel edges in F for each $i = 1, \dots, h$. Let ϕ be the mapping from V to U defined by $\phi(v_i) = u_i$ for $i = 1, \dots, h$ and $\phi(v) = u_0$ for $v \in V \setminus \{v_1, \dots, v_h\}$. Then C and ϕ form a cactus for G as described in Theorem 30.

Suppose instead that $\bar{\lambda} \leq \delta$. Then $\bar{\lambda}$ is also the edge connectivity λ of G . We then apply the cactus algorithm of Gabow [8] to \bar{G} . In $\tilde{O}(\bar{\lambda}\bar{m}) = \tilde{O}(m)$ time, it produces a cactus $C = (U, F)$ of \bar{G} and a mapping $\bar{\phi}$ from \bar{V} to U as in Theorem 30. Next we turn $\bar{\phi}$ into a mapping $\phi : V \rightarrow U$ from the original vertex set V by reversing the contractions from Theorem 1, that is, if v got contracted into the super vertex \bar{v} , then $\phi(v) = \bar{\phi}(\bar{v})$. Now we have a cactus representing some min-cuts of G , including all non-trivial min-cuts of G . If $\bar{\lambda} < \delta$, then there are no trivial min-cuts of G , and then our cactus C is the final cactus for G .

Finally, if $\bar{\lambda} = \delta$, there may be some trivial min-cut of G that are not yet represented. The min-cut around a min-degree vertex v is represented if and only if there is a vertex $u \in U$ such that $\{v\} = \phi^{-1}(u)$ and u has only two incident edge. If this is not the case, let $u = \phi(v)$. To include the min-cut around v , we introduce a new vertex u' in U and set $\phi(v) = u'$. The only neighbor of u' is u and we add two parallel edges between them to F . This adds the desired trivial min-cut but no other cuts to the cactus representation. We repeat this process for all min-degree vertices whose min-cut is not yet represented. Now C and ϕ is a cactus for G as described in Theorem 30. Adding the trivial min-cuts took $O(m)$ time, so the whole construction time is $\tilde{O}(m)$. This completes the proof of Theorem 31. ■

7 Acknowledgments

We would like to thank Hal Gabow and Yuzhou Gu as well as anonymous STOC referees for patiently reading earlier versions of this paper, pointing out issues and coming with useful suggestions for the presentation.

References

- [1] R. Andersen, F. R. K. Chung, and K. J. Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.
- [2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *J. Comput. Networks and ISDN Systems archive*, 30:107–117, 1998.
- [3] P. Chalermsook, J. Fakcharoenphol, and D. Nanongkai. A deterministic near-linear time algorithm for finding minimum cuts in planar graphs. In *Proc. 15th SODA*, pages 828–829, 2004.
- [4] E. A. Dinits, A. V. Karzanov, and M. V. Lomonosov. On the structure of a family of minimum weighted cuts in a graph. *Studies in Discrete Optimization*, pages 290–306, 1976.
- [5] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal of Computing*, 4:507–518, 1975.
- [6] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

- [7] A. Frank. On the edge-connectivity algorithm of nagamochi and ibaraki. *Laboratoire Artemis, IMAG, Universitk J. Fourier, Grenoble*, 1994.
- [8] H. N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In *Proceedings of the 32nd Annual Symposium on the Foundations of Computer Science*, pages 812–821, 1991.
- [9] H. N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comp. Syst. Sc.*, 50:259–273, 1995. Announced at STOC’91.
- [10] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society of Industrial and Applied Mathematics*, 9(4):551–570, 1961.
- [11] J. Hao and J. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *Journal of Algorithms*, 17(3):424–446, 1994. announced at SODA’92.
- [12] M. Henzinger and D. Williamson. On the number of small cuts in a graph. *Inf. Process. Lett.*, 59(1):41–44, 1996.
- [13] J. Holm, K. Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge and biconnectivity. *J. ACM*, 48(4):723–760, 2001. Announced at STOC’98.
- [14] D. R. Karger. Random sampling in cut, flow, and network design problems. *Math. Oper. Res.*, 24(2):383–413, 1999. Announced at STOC’94.
- [15] D. R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000. Announced at STOC’96.
- [16] D. R. Karger and D. Panigrahi. A near-linear time algorithm for constructing a cactus representation of minimum cuts. In *Proc. the Twentieth ACM-SIAM Symp. on Discrete Algorithms*, pages 246–255, 2009.
- [17] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4), 1996. Announced at SODA’92 and STOC’93.
- [18] A. Karzanov and E. Timofeev. Efficient algorithm for finding all minimal edge cuts of a nonoriented graph. *Kibernetika*, 2:8–12, 1986. translated in *Cybernetics*, (1986), pp. 156-162.
- [19] K. Kawarabayashi and M. Thorup. Deterministic global minimum cut of a simple graph in near-linear time. In *Proc. 47th STOC, to appear*, 2015.
- [20] D. W. Matula. Determining the edge connectivity in $o(mn)$ time. In *Proc. the 28th Annual Symp. on the Foundations of Computer Science*, pages 249–251, 1987.
- [21] D. W. Matula. A linear time $2 + \epsilon$ approximation algorithm for edge connectivity. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, pages 500–504, 1993.
- [22] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.
- [23] H. Nagamochi and T. Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM Journal of Discrete Mathematics*, 5(1):54–66, 1992. Announced at SIGAL’90.

- [24] H. Nagamochi and T. Ibaraki. Linear time algorithms for finding a sparse k -connected spanning subgraph of a k -connected graph. *Algorithmica*, 7:583–596, 1992.
- [25] V. D. Podderyugin. An algorithm for finding the edge connectivity of graphs. *Vopr. Kibern.*, 2:136, 1973.
- [26] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, NY, 2003.
- [27] M. Stoer and F. Wagner. A simple minimum cut. *J. ACM*, 44:585–591, 1997. Announced at ESA'94.